User Context Centric Models in A Knowledge Advantage Machine

Luyi Wang[1], Ramana Reddy[1], Sumitra Reddy[1] & Asesh Das[2]

***Abstract*—User centric interaction is an essential requirement in many types of web related applications. Always a basic expectation from the user is that the web interaction must be self-organizing and evolving in an organized pattern as the user interest progresses. This paper addresses such an expectation with an ontology pattern based Knowledge Advantage Machine [KAM]. It illustrates how this KAM system leads to the determination of a reliable context as the knowledge discovery is taking place.**

Index Terms*—*Knowledge Advantage Machine, user profile, ontology, patterns, context awareness, user context centric.

I. Introduction

Quality of Service (QoS) is directly related to addressing the user preference [1]. In the midst of voluminous user data accumulation and sophisticated user models, finding a stable QoS providing system is a challenging issue. Aiming at that, ideas were proposed for automating the capture of user preferences, which will provide flexibility on related services [2], [3]. Most of these services needed intermediate communication protocols to configure a solid user base, which is the core service for all upper layers. Because of this, ontology engineering becomes a crucial element in all core services at the user level. The basic premise of ontology operation is to have a sharable taxonomy, which follows certain logic reasoning. In this context, we have introduced the Knowledge Advantage Machine (KAM) model for addressing the knowledge sharing problems [4]. The basic knowledge unit in KAM, dubbed as JAN, can be extracted from various sources such as documents, web pages or emails. After the extraction, it becomes a logical unit, referring and linking other JANs to it. While acting as the basic reusable unit in the resource-oriented architecture (ROA), the JAN plays a strong role in representing knowledge. JAN may be viewed as a revision of the learning object metadata (LOM) standard [11]. According to the definition of LOM, it needs to contain necessary components, such as annotations, references, and categories. These components are essential in the KAM model while dealing with knowledge processing and discovery. From a user's perspective, the knowledge units, JANs are unique, reflecting user interest and preferences. Annotations and references labeled on a JAN emphasize its category and relationships in user's ontology. Thus, JAN not only serves as a knowledge unit, but also contains meta-information describing its conceptual meaning and the user's view on a certain area of interest. We can also state that a JAN is only meaningful when combined with the user perspective. In reality a standard user usually crosses over multiple domains. For instance, people always have family domains along with work domains. In the KAM definition, one KAM resides on only one domain in which a user's ontology manages resource classifications and their relationship. Typical user ontology mainly contains two parts: taxonomies consisting of JANs, and taxonomies reflecting relationships between JANs.

This work was supported in part by an endowment from the Verizon Foundation.

[1]Luyi Wang, Ramana Reddy and Sumitra Reddy are with KaMlab, Lane Department of Computer Science & Electrical Engineering West Virginia University, Morgantown, WV 26506, USA

[2] Asesh Das is with School of Business & Computer Technologies Pennsylvania College of Technology (Penn State) Williamsport, PA 17701, USA

II. USER PROFILE BASED ONTOLOGY

A user profile usually contains two necessary parts. One is the basic user information, and the other is the user preference. The former is unique for every user. But the latter part is shared among people who share a common interest. It also plays an important role in our knowledge engineering model. In KAM, the user preference is represented by the user ontology. We define the user preference as a class of basic taxonomies. Along with these two aforesaid user profile parts, we also define user behavior. A typical user behavior describes the user's action on a knowledge unit. For example, a user browses a particular web page or reads a specific document. All these user actions are recorded as user logs. By analyzing the user's actions and the knowledge units related to them, the KAM organizing agent (OA) generates user behavior rules. Based on these rules, the KAM helps a user to find JANs that are potentially interesting and relevant. The KAM organizing process focuses two aspects: (i) A new JAN classification - when a new information object JAN comes in, the OA tries to find a suitable taxonomy where this JAN can reside in. This aspect is accomplished by executing queries on user ontology. The query results contain taxonomies with high possibility of occurrences. (ii) Find related JANs - the contents of the most browsed JANs are extracted and a query containing this content is submitted to the KAM discovery agent (DA). The query will return the JANs with the highest content similarities. Traditional information retrieval methods rely on keywords to describe the content of an information unit. In KAM, the information unit, JAN is organized into a taxonomic structure. This method allows us to transform the query into two parts: one part is the traditional search on the knowledge units and the other is a concept search that returns taxonomies in which the knowledge unit resides. To implement these, we applied several statistical probability models to map phrases into concepts. For each JAN, its original resource content was extracted using the Vijjana key phrase extraction algorithm (VKE) [12][13], and a similarity test was performed on all user taxonomies to determine its underlying concepts. If no similar concept was found, a universal similarity test was performed on a global ontology - explained in the next section. According to Hele mai Haav et al [5], approximately 3,000 terms will cover all general concepts for a specific domain. For a typical user, it is possible to use a finite taxonomy to cover all the domains he/she crosses over.

III. GLOBAL ONTOLOGY

To construct a user profile with ontology information, we needed a large reference data repository. Open Directory Project (ODP) came to be the final choice after we reviewed several data sources [5], [6]. The ODP is regarded as one of the largest taxonomy stores for web directories. The taxonomy is organized in a hierarchical structure. It has become customary to use ODP as a main reference source and the top three levels of taxonomy are used as references promoting the ontology hit accuracy. In KAM framework, we also used taxonomies in the first three levels as our global concept set. Our purpose was to construct a universal ontology. We first analyzed the structure of the ODP data. The ODP data contains two parts. One is its hierarchy structure and the other is a large RDF file containing all links and descriptions of their hierarchy structure. To convert it to our global ontology, we reorganized all into one unit. In KAM, the ontology is defined as a set of taxonomies with two features: (i) It has siblings on the same level. (ii) For every node in the hierarchy, we can find corresponding items in the resource description (RDF) file. The RDF item is usually a bookmark link with its self-description. We could map the ODP hierarchy node as our universal ontology taxonomy, and the RDF item as the knowledge unit JAN in the universal ontology. Thus, the universal ontology defined in KAM contains relationships between its parents and siblings, which were also taxonomies. Figure 1 is a partial view of the universal ontology. The first level contains 14 taxonomies. The second level contains 517 taxonomies. And the third level contains 6056 taxonomies.

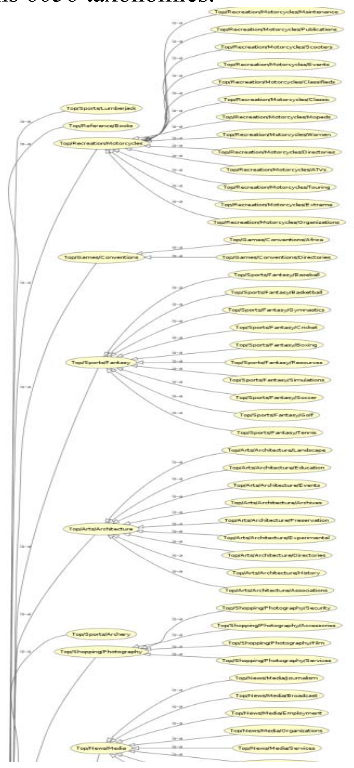

Top/Recreation/Motorcycles/Maintenance
Top/Recreation/Motorcycles/Publications
Top/Recreation/Motorcycles/Scooters
Top/Recreation/Motorcycles/Events
Top/Recreation/Motorcycles/Classifieds
Top/Recreation/Motorcycles/Classic
Top/Sports/Lumberjack
Top/Recreation/Motorcycles/Mopeds
Top/Reference/Books
Top/Recreation/Motorcycles/Women
Top/Recreation/Motorcycles
Top/Recreation/Motorcycles/Directories
Top/Recreation/Motorcycles/ATVs
Top/Recreation/Motorcycles/Touring
Top/Recreation/Motorcycles/Extreme
Top/Recreation/Motorcycles/Organizations
Top/Games/Conventions/Africa
Top/Games/Conventions
Top/Games/Conventions/Directories
Top/Sports/Fantasy/Baseball
Top/Sports/Fantasy/Basketball
Top/Sports/Fantasy/Gymnastics
Top/Sports/Fantasy/Cricket
Top/Sports/Fantasy/Boxing
Top/Sports/Fantasy
Top/Sports/Fantasy/Resources
Top/Sports/Fantasy/Simulations
Top/Sports/Fantasy/Soccer
Top/Sports/Fantasy/Golf
Top/Sports/Fantasy/Tennis
Top/Arts/Architecture/Landscape
Top/Arts/Architecture/Education
Top/Arts/Architecture/Events
Top/Arts/Architecture/Archives
Top/Arts/Architecture/Preservation
Top/Arts/Architecture
Top/Arts/Architecture/Experimental
Top/Arts/Architecture/Directories
Top/Arts/Architecture/History
Top/Arts/Architecture/Associations
Top/Shopping/Photography/Security
Top/Sports/Archery
Top/Shopping/Photography/Accessories
Top/Shopping/Photography
Top/Shopping/Photography/Film
Top/Shopping/Photography/Services
Top/News/Media/Journalism
Top/News/Media/Broadcast
Top/News/Media/Employment
Top/News/Media/Organizations
Top/News/Media
Top/News/Media/Services
is-a

Fig. 1. Global Ontology

IV. CONSTRUCTING USER ONTOLOGY

In the KAM, user ontology is used to manage the user knowledge network containing all user knowledge units. Here we extract JANs from three knowledge unit sources: User Email, User File System and the User bookmarks.

A. JAN Abstraction

As we discussed earlier, a JAN is abstracted from various sources. Here, JAN "reference" denotes the source file. When a new JAN is brought into view, its reference needs to go through three operations:

- Extraction of content.
- Preprocessing - removing stop words and stemming.
- Full Text and VKE processing

The first step is to extract the content. For the KAM user file system, the current implementation supports three basic types of textual documents: text, pdf and MS Word. For the KAM user bookmarks, we parsed the bookmarked webpage and distilled its content. For the KAM Email, we concatenated its subject and body to form the content.

In the stop-word removal process, we maintained a large stop-word corpus. The extracted content was parsed with this corpus and the leftover part did not contain any word in this corpus. The remaining part after this step was passed to a stemming process that removed token suffixes and recovered the base, or stem, of the word. Step three had two options: FULL TEXT and VKE processing. The FULL TEXT processing uses all results from step two for key phrase indexing. The VKE processing only builds an index upon the key phrases generated by the VKE algorithm. The phrases used in an index were considered to be annotations of the JAN. After these three steps were performed, a JAN was created.

B. User File System

To build user ontology from the user's file system, there is a need to traverse through all user folders and files contained therein. Since the folders have hierarchical structures, we mapped them to taxonomies in the user's ontology. Correspondingly, files were regarded as knowledge units under the taxonomy.

We implemented two interfaces: the "KAM File System" and the "File Descriptor". The "KAM FileSystem" (KFS) managed the folder structure. To create a KFS, the users must specify the KFS root path. The KFS has the capacity to navigate through its subfolders. It can recursively go deeper to the lowest level, and report the whole structure. During this process, it can call a "File Descriptor" process, which is responsible for generating file meta information and extracting the file content. The file content and the file information were used by the JAN creation process.

The FULL TEXT processing in KFS has two streams: the Apache Lucene or our own method (which is explained in the forthcoming sections). A generated user ontology upon file system is shown in the Figure 2.

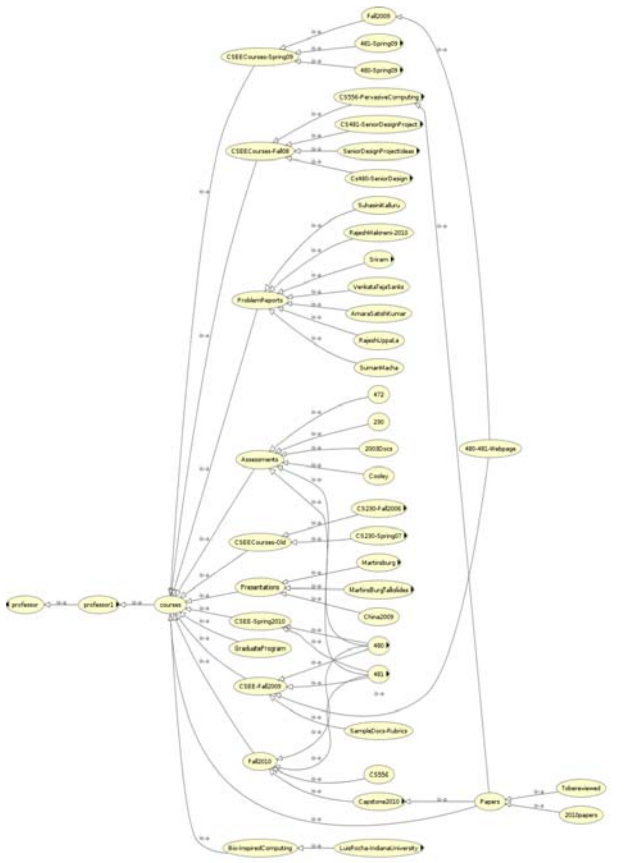
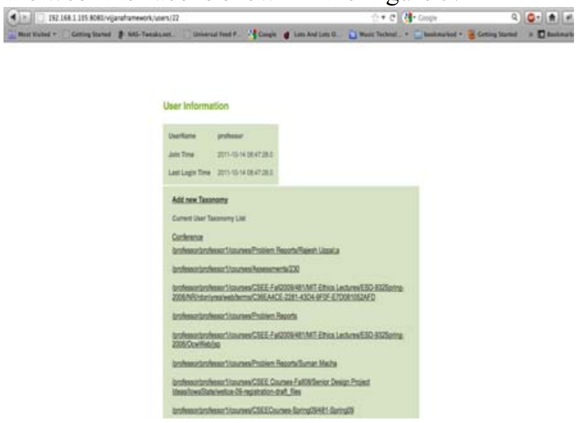


Fig.2 User ontology.

C. *User Bookmark System*

The user bookmark system is a web interface that allows users to save their own bookmarks. In Devalapalli et al [7] we illustrated old Firefox plug-ins to help the user through the JAN creation progress. The process has been simplified in our new web interface. In addition to that, from the web interface a user can create a user space to store all the bookmarks. Also, this user space can be shared with the KAM File System. From the web interface, a user can browse the ontology generated from KFS. The JAN creation process is initialized when a new link is added to user space. The discovery agent running as a part of web service automatically extracts contents from a linked source. The content will go through the same process as in the JAN abstraction. However from the web service, FULL TEXT is only supported in our own version at this moment. Ontology of the web interface is shown in the Figure 3.


192.168.1.105:8080/vijanaframework/users/22

User Information

UserName	professor
Join Time	2011-10-14 08:47:28.0
Last Login Time	2011-10-14 08:47:28.0

Add new Taxonomy

Current User Taxonomy List

Conference
/professor/professor1/courses/Problem Reports/Rajesh Uppala

/professor/professor1/courses/Assessments/230

/professor/professor1/courses/CSEE-Fall2009/481/MIT Ethics Lectures/ESD-932Spring-2006/NR/rdonlyres/web/terms/C36EA4CE-2281-43D4-9F3F-E70081052AFD

/professor/professor1/courses/Problem Reports

/professor/professor1/courses/CSEE-Fall2009/481/MIT Ethics Lectures/ESD-932Spring-2006/OcwWeb/js

/professor/professor1/courses/Problem Reports/Suman Macha

/professor/professor1/courses/CSEE Courses-Fall08/Senior Design Project Ideas/IowaState/welcs-09-registration-draft_files

/professor/professor1/courses/CSEECourses-Spring09/481-Spring09

Fig3. User Ontology on Web interface

V. Context Awareness

Another important part of the user profile is user behavioral model. Our assumption in KAM is that we can use a finite number of taxonomies to represent a user's knowledge domain. Based on this assumption, all user behaviors are converted into activities crossing over the taxonomies. To determine the user preferences, we can generalize a finite number of rules by monitoring the transitions happening in between taxonomies. For instance, when off of work, a user who has a strong interest in sports may spend more time reading sports news than reading financial news. For this particular use, when the transition from work to news occurs, KAM shall promote the sports news ahead of financial news. In KAM, the taxonomy priority is evaluated by the taxonomy interest score. The interest score is affected by two factors: total hit number and taxonomy size. The first one is the number of times the user browsed the taxonomy and the second one denotes how many JANs are related to this taxonomy. We kept updating the taxonomy interest score when the user browses the taxonomy or adds new JANs into it. The interest score is calculated as:

$$I(t_i) = total\_hit\_number / taxonomy\_size$$

$I(t_i)$ stands for the user interest in taxonomy i, *taxonomy_size* is the number of JANs in this taxonomy. As has been stated already, a user behavior model forms our basis for context awareness. To detect in which context a user resides, KAM uses methods that fall into two categories: the timeline and the knowledge hit statistics. Before proceeding to explain our context awareness model, we need first to define what a context is. From its semantic meaning, a context is where the user is. In our daily life, a context can be a restaurant where people are having dinner. When a person reads a book, the context is the paragraph that is engaging the reader. In a knowledge network, for example, in the ODP project, the context is the branch where people click links. In the KAM model, we defined the context as the current ontology on which the user is working. Recall that in the user behavior model, a transition is like a taxonomy switch. Here we can simply define the current context as the current state, which is a taxonomy. Therefore, one of the other taxonomies would possibly become the next state. Context={Current, {Next}}.

A. *Timeline Context Awareness*

In the KAM web interface, every operation on a taxonomy and JAN is recorded as a user history entry. When a new taxonomy or JAN is created, its creation time is logged. In addition to that also a record is kept when the user browses the taxonomy or JAN. The track data is used in the calculation of the interest score. If a particular user, in a certain time period everyday, always browses certain taxonomy or related JANs, the KAM would mark this time period with this taxonomy information, and set it as the user context for this time slot.

Using this method, we first sectioned the day off as being part of one of two classifications: either active or inactive. The basic time unit of timeline mode was one hour. The inactive periods were time units without any user activity. For a certain time phase of enough length, such as a week or a month, the user's activity could be categorized by these two periods. For the user's active period, we could detect the taxonomy boundary if we already knew the taxonomies the user owns. Based on the user's activities for taxonomies, we could calculate the probability for each taxonomy on time phase, P(t_i| time j), and then choose the highest one as the user's context.

B. *Interest Driven Context*

Each taxonomy has an interest score calculated as explained in the user behavior model. For all taxonomies, each concept has a factorial value between zero and one to describe its importance.

$$PI(T_i) = I(T_i)/ \sum I(T_i)$$

By this weight value, we also can predict the most probable next state. This memory-less sequence forms a stationary Markov chain. The transition probability for a user moving from one taxonomy to another one, $P(t_{\{next\}}|t_{\{prior\}}) = P(t_{\{next\}})*P(t_{\{prior\}})$, can be derived from a transition matrix. It is more likely that a user would move from current state to taxonomy with the largest probability. According to the transition matrix, we can form a priority queue that stores a certain number of taxonomies with the highest probability. The next state of context is selected from this queue. Every transition of context switch would update the interest score and consequently update $PI(T_i)$. Generally, for a small user knowledge network with a low average hit number, the update operation would not be costly.

VI. Classification of JANs

Up to this point, the KAM has enough information to provide the user a suggestion on how to organize his/her knowledge units. The KAM shares a universal ontology built from the ODP data. Additionally, the KAM owns its own user ontology, and user behavior history. All these are used when JANs are classified.

As we mentioned already, in the JAN abstraction process, the JAN annotations are also created and would be used as index phrases. Therefore, a JAN can be represented by a word vector $j=\{w_0,w_1,\ldots,w_n\}$. The corresponding taxonomy can be represented as a class containing all these JANs. For all KAMs, they all share a common global ontology in which the taxonomies and the knowledge units are all the same. For a specific user's KAM, taxonomies are created by that user. Even though JAN annotations are either FULL TEXT or generated by KAM, they can be edited by the user. When a new JAN is added into a user's ontology, the methodology of selecting a suitable taxonomy used for both global and personal ontology becomes different. However, there is a cohesion existing in the training data processing of the global and the personal ontology.

A. *Generating the training data*

The term frequency–inverse document frequency (TF*IDF) [10] is a popular technology used in text classification. From our point of view it is essentially:

$w_{ij} = tf_{ij} * idf_i$, where
tf_{ij} = term weight, &
idf_i=log((JAN training set size)/number of JAN containing t_i)

Here the term weight of a JAN annotation is calculated using the augmented normalized term frequency. The augmented normalized term frequency is described as:

$$JAN\ (tf_w) = 0.5 + 0.5* tf_w/tf_{max}$$

where, tf_w is the occurrence frequency and tf_{max} is the maximum term frequency in JAN annotations. Here the normalization process, tf_w/tf_{max}, removes the dependency of classification results on annotations length [8]. Also it ensures the correctness of using the VKE algorithm result for JAN annotations.

To calculate the weight of each annotation using the TF*IDF method, we needed to specify the training set. In the global ontology, we used the top three taxonomy levels as a training set. In user space, we employed all user created taxonomy data as the training set. A taxonomy word vector consisted of the sum of all its contained knowledge units the JANs' word vectors have. We used V_t to denote a taxonomy word vector, and J_t for JAN word vector. And we have, $V_T=\{V_j|$ all j in }.

Once we had the training set ready, for any given JAN, we could calculate its TFIDF weight. The TFIDF weight of a JAN is described as:

$$TFIDF(jan)=\{\sum w_i|where\ w_i = tf_i*idf\}$$

Consequently, the taxonomy weight is described as:

TFIDF{Taxonomy} = ∑ JAN, where JAN belongs to same taxonomy.

To remove the classification error caused by mismatched vector lengths, we also applied a normalization process on taxonomies. There are many normalization methods. Cosine normalization is the most commonly accepted one. The cosine normalization is described as:

$$CN(V) = (w*_1,w*_2,\ldots,w*_n)\ \ where,$$
$$w*_i = w_i / SQRT(\sum (w_i* w_i))$$

B. *The Similarity between JAN and taxonomy*

In order to determine a suitable taxonomy for an incoming JAN, we used cosine similarity to get the user suggestions. A new JAN is represented by a vector J= $\{w_0, w_1,\ldots, w_n\}$, where w_i is the term that appeared in the JAN annotation. The taxonomy vector is comparatively larger than a user taxonomy vector. Let T= $\{w_0, w_1,\ldots, w_m\}$, represent the taxonomy and w_j stands for the term that appeared in it. For these two vectors, we formed their weight vectors using the term's TFIDF value. To calculate their cosine similarity, the two vectors' lengths must be equal. Then we needed to construct two equal length vectors. We first merged the two word vectors together to form a new vector, and then we used this method to construct the weight vector. And for every word that was missing in the original word vector, we filled its weight with 0. After this, we could calculate the cosine similarity. The cosine similarity of these two vectors can be expressed as:

$$cosine(t_i,JAN(j)) = \sum w_{ik}* w_{jk},$$

where, w_{ik} indicates the TFIDF weight of term k appearing in the taxonomy i.

The final cosine similarity value is between -1 and 1. The closer is the absolute value to 1, the more similar the two vectors are. Consequently, we could determine whether a particular JAN is similar to the taxonomy. In addition to this value, we also considered the interest hit value. In the final suggestion rank, the weight consisted of two parts: the cosine similarity result and $PI(t_i)$

$$R(T) = Cosine(T,j)*PI(t_i)$$

Based on this rank value, we provided suggestions to the user for the right taxonomy of this JAN.

C. The Relationship between local taxonomy and global taxonomy

In the KAM, one important part is the communication capacity between different users. All users should know who shares interests with them. This part is bridged with the help of the global taxonomy. While a new JAN is added into user taxonomy, the KAM also performs the similarity test upon global taxonomies. The relationship between different users is established for a JAN sharing the same annotation with a global ontology.

VII. KNOWLEDGE DISCOVERY PROCESS

One user, at a given point in time, should only reside in only one context. As we have defined already, the context provides the current taxonomy and the next series of possible taxonomies. For each user, his/her ontology shares part of the universal ontology. With this feature, we can regard users who share the same ontology as a community. For instance, professors doing research in computer science should share the ontology concerning computer science. All communities share the universal ontology. The Figure 4 illustrates this idea.

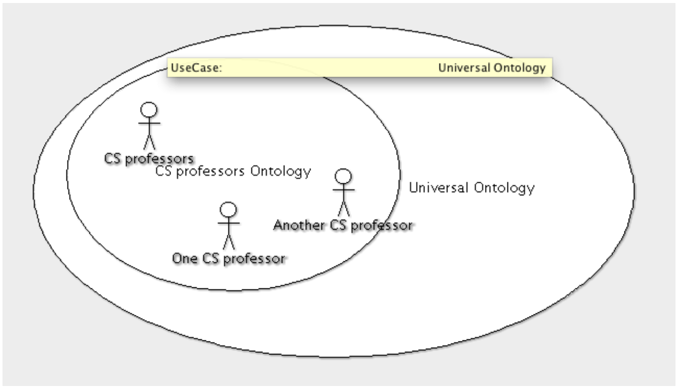


Fig. 4. Discovery Process

This also leads to the knowledge discovery process, which is a three-step procedure. We named this procedure as "Call it Once". The discovery first happens locally, in the user context. Then it expands to communities, where the user resides in the same ontology, and then explores the universal ontology. Knowledge discovery can be initiated by a user in a certain context or by an agent during context switching. No matter in which way this occurs, it is performed by queries upon taxonomy. A typical query constitutes a set of phrases.

A. The local Knowledge discovery process

The local search is confined within the user taxonomies. For a given query, the cosine similarity described earlier can still be used. If the search is confined to a particular taxonomy range, for every JAN, we could perform a similarity test and the JAN with the largest value was returned as the query's result. However, if the search is not confined to any taxonomy, the similarity test needs to be performed between the query and all taxonomies. The method is already illustrated in the cosine similarity part. We would construct a bigger vector and fill the missing term weights with 0's and perform cosine similarity again. The process acting on the returned taxonomies was similar to what has been done for an individual taxonomy.

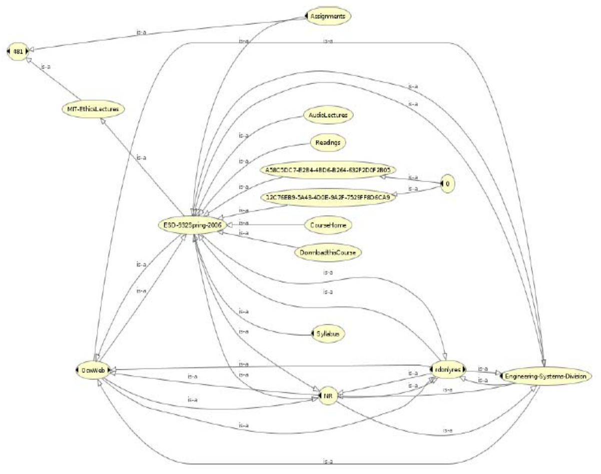


Fig. 5. The Local Search

The Figure 5 describes a local search belonging to a professor. When the professor is preparing the course CS 481, she finds a part of her ontology on MATH courses that was saved before.

B. The Community Knowledge discovery process

Since people in the same community share the same ontology, we could use the collaborative filtering (CF) technique [9] to recommend JANs to a user. There are two types of collaborative filterings: user based CF and item based CF. Recall in KAM model, in order to eliminate the problem of organizing and consistency checks, we applied ROA architecture which required that all items be uniquely identified. For an original resource, it is abstracted into a JAN and added into user's knowledge network. After this step, it became unique in the whole knowledge network. We could not directly apply the CF technology upon the JANs. There are two methods to solve this problem. The first one is the rudimentary one that uses the JAN's reference, the original resource, as our item. The other one uses a keyword to replace the JAN as the comparison item. For the first method, we can construct the user-item matrix, in which item's value was the hit number of the JAN.
Then,

$$UI_{ij} = \begin{cases} \text{hitnumber}_i & \textit{if item i also appears in User j's knowledge network} \\ 0 & \textit{Otherwise} \end{cases}$$

Once we had this matrix, we could use the adjusted cosine similarity to compare the two JANs. The JANs with highest similarity should catch the user's eye. The performance difference between the item-based and the user-based methods depends on the sparsity of the matrix that was built. If the matrix is sparse, the user-based performance should be poorer than the item-based one.

For the second method, we used a keyword to replace the JAN, so the user-item matrix was formed in the following manner. Here are the keywords as a set of JANs :

$$UI_{ij} = \begin{cases} 1 & \textit{if item i also appears in User j's knowledge network} \\ 0 & \textit{Otherwise} \end{cases}$$

We calculated the similarity between the two keywords and recommended JANs, sorted with the highest similarity keyword for the user.

C. The Universal Knowledge discovery process

The final step in "Call it Once" process is to search the global ontology. Here the query is without any user preference. For taxonomies in global ontology, we can perform a local search on user's related taxonomies, and regard the returned JANs as a compensation of the result from the global search and the community search. So the overall discovery process can be viewed as in the Figure 6.

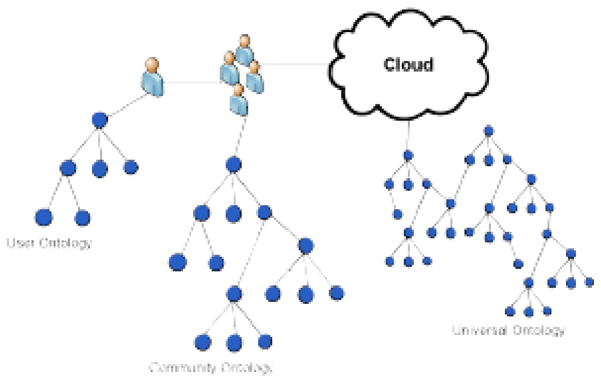


Fig. 6. The Global Search Process

VIII. CONCLUSION & FUTURE WORK

In this paper, we illustrated our approach in building up a user profile from ontology. The focus was on building user ontology from global ontology, personal file system and the bookmark system.

Based on the ontology we have built, two context awareness models were presented in discovering user preferences. In the KAM, we used these two models to predict the next user states and provided the state related knowledge. The knowledge discovery process is what we called "call it once" in the KAM.

It divided the search into three phases. The first one is the searching on local. It mainly focuses on finding the accurately matched local files or taxonomies. The second one is the searching on community, which is based on the CF algorithm to provide recommendations. The third one is a global search without any preferences. We are still fine tuning the context model. Also, it would be necessary to increase accuracy of the CF algorithm to ensure scalability to real-world problems.

REFERENCES

[1] Altmann, J, and P Varaiya. "INDEX project: user support for buying QoS with regard to user's preferences." 1998 Sixth International Workshop on Quality of Service IWQoS98 Cat No98EX136 (2005) : 101-104
[2] Lisa Gottlieb and Juris Dilevko. 2001. User preferences in the classification of electronic bookmarks: implications for a shared system. J. Am. Soc. Inf. Sci. Technol. 52, 7 (May 2001),
[3] User-preference-based service selection using fuzzy logic,Zhengping Wu Mu Yuan , Network and Service Management (CNSM), 2010 International Conference
[4] Luyi Wang, Ramana Reddy, Sumitra Reddy, and Asesh Das, A Context Centric Model for building a Knowledge advantage Machine Based on Personal Ontology Patterns, in WorldComp 2011(SWWWS'11), pp 99-105
[5] Hele mai Haav and Tanel lauriLubi. A survey of concept-based information retrieval tools on the web. In In 5th East-European Conference, ADBIS 2001, pages 29-41, 2001.
[6] Joana Trajkova and Susan Gauch. Improving Ontology-Based User Profiles, 2004.
[7] S. Devalapalli, R. Reddy, L.Wang, and S. Reddy. Markup and Validation Agents in Vijjana- Pragmatic Model for Collaborative, Self-organizing, Domain Centric Knowledge Networks. In WEBIST, pages 263-269, 2009.
[8] Verayuth Lertnattee and Thanaruk Theeramunkong. Effect of term distributions on centroid based text categorization. Inf. Sci. Inf. Comput. Sci., 158:89-115, January 2004.
[9] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th international conference on World Wide Web (WWW '01). ACM, New York, NY, USA, 285-295.
[10] Gerard Salton, Christopher Buckley, Term-weighting approaches in automatic text retrieval, Information Processing; Management, Volume 24, Issue 5, 1988, Pages 513-523, ISSN 0306-4573, 10.1016/0306-4573(88)90021-0.
[11] 2002 Learning Object Metadata (LOM) Standard Maintenance/Revision, IEEE 2002
[12] Luyi Wang, Ramana Reddy, Sumitra Reddy and Asesh Das, Keyphrase extraction algorithm in Knowledge Advantages Machine, Draft version,
[13] Luyi Wang, Ph.D. dissertation 2012, West Virginia University-Library Archives