A Context Centric Model for Building a Knowledge Advantage Machine Based on Personal Ontology Patterns

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

[1]SIPLab, Lane Department of Computer Science & Electrical Engineering West Virginia University, Morgantown, WV 26506, USA
[2]School of Business & Computer Technologies
Pennsylvania College of Technology (Penn State) Williamsport, PA 17701, USA

Abstract - *A knowledgeable person, in old times, has been regarded as a saint who knows everything without stepping out of house. Current technologies provide us with the ability of acquiring knowledge as one saint did before. Today, overloading of information always results in the quest for efficiently taking possession of useful information. With the advent of pervasive computing, information gathering is happening in real time. It has become possible that a collaborative knowledge network is constructed by utilizing little resource, but accurate under a carefully established environment. In this paper, we propose a new architecture, knowledge advantage machine (KaM), that helps people to construct a knowledge network. And we would like to apply a term, "call it once", into a novel type of knowledge discovery method. A typical User of KaM machine covers only one particular domain. The Domain information consists of a user defined ontology, a set of user collection of knowledge unit, we called them JANs, and some necessary tools to facilitate the knowledge discovery process. The "call it once" model transforms the traditional knowledge discovery process to a few basic simple steps (three steps).*

Keywords: knowledge discovery, knowledge network, knowledge advantage machine, personal ontology, artificial intelligence.

1 Introduction

The internet has indeed changed the habit of people searching of knowledge. We often talk on how Google or Facebook has changed our life. In reality however, knowledge itself is not provided by search engines. Only a consolidated body of related information is provided following a few key words. As a sequel, too much uncategorized information is delivered, causing wastage of time and frustration in finding a typical set of knowledge. For instance, people, who intent to learn programming on smart phones, may expect a body of knowledge different from those who are trying to buy a new phone. This obviously amounts to saying that knowledge discovery is not some straight forward disciplines, there are some other crucial factors affecting it. When portable devices have become prevalent, in a short time domain-specific, on-demand call for knowledge is a mandate. This brings the need for making newer knowledge discovery methods developed, which will be quick, to-the-point and will require little effort from the user. Such an expectation simply means that knowledge discovery process should be context-centric. It has been well-known that by monitoring and analyzing user behavior a detailed personal profile can be built, which however, often becomes very general and devoid of contextual bearing. This problem is overcome merging ontology and domain knowledge. A user ontology is regarded as speculations on concepts and relationships among user's particular domain taxonomy. To provide an on-demand information, this domain taxonomy with information can be investigated leading to providing one particular knowledge of a particular user's particular interest. In this paper, we are proposing a new architecture which is suitable for building up a user context centric knowledge network. The architecture is named as Knowledge Advantage Machine (KaM). The KaM is aimed in effectively helping people to find the right information with lesser effort. The paper is organized as follows. The section 2 narrates previous research in this line. The section 3 focuses on the concept of the KaM Machine and Vijjana model. The section four is about the current approach in constructing a user context centric mode. The section 5 concludes the work.

2 Related research

In the last two decades, people have spent an enormous of time and effort in the science and technologies of knowledge discovery. Gruber's 1993 work [1] on the techniques of developing ontology has become a classic research. It was demonstrated that ontology can be treated as agreements on knowledge sharing leading to reusable knowledge component and knowledge based services. In 1995 Chen and Ng [2] proposed two new algorithms on knowledge discovery using weight factors on knowledge relationships measuring similarities in between. In 2005, the concept of semantic desktop emerged which illustrated the technology for personal workspace and semantics-enabled desktop [3]. In 2006, semantic desktop 2.0 discussed about how to organize personal user's resources relying on metadata [4]. In 2008, people began to explore semantic web from mobile device perspectives [5]. In 1995, proposal [6] was made to build up profiles by monitoring user webpage browsing history. Starting from 1999, techniques on building personal profile started to categorize information with abstract structure. Proposals to generate user interest listing [7, 8], and methods of organizing information in to hierarchically evolving information evolved. Some leads on context awareness were introduced. Based on user feedbacks Researchers [9,10] tried to define what circumstances the user is in, and to categorize the user interests in to concepts and forms hierarchy. The use of Open Directory Project [11] as source ontology, and using the traditional vector space modeling methodologies were delivered [12].

3 KaM concept and the Vijjana model

In this paper, we proposed our knowledge discovery approach called the “Knowledge advantage Machine" which focuses on binding ontology pattern from user activities and discovering knowledge based on user context. “Knowledge advantage Machine" term is derived from “Mechanical advantage". The first two words, “Knowledge Advantage", is stressing the knowledge importance. In the industrial era, the mechanical advantage helped people to promote productivity in utilizing different mechanical tools. When time changed to the current era--the information era, the fast accelerating information processing power not only forced people to learn more but also required people to alter unuseful information. The last word “machine" relates to architecture. KaM consists of many components which work collaboratively. To address the context aware capacity, we would emphasize that our "KaM" should be applied only on one domain at one time. Here is how we introduced the KaM.

1. One KaM should be defined within a particular domain, enriching itself with domain information.
2. Within one domain, it can contain more than one KaM.
3. One KaM should reside in only one domain, but it can interactively communicate with other KaM residing on any other domain.
4. Ontology information of one KaM constitutes parts of ontology of the domain it resides in.
5. One KaM consists of at least one agent working on knowledge residing in it.

Here is a scenario to illustrate the aforesaid KaM architecture: The schedules for Jim on Tuesday is mostly depending upon the research Jim carrying on. He is writing one proposal for a certain science/technology funding. He communicates with professors via email in other universities to collaborate on the same proposal. All of them spend a great amount of time in discussing several innovative ideas that utilize some theories abstracted from biological phenomenology in the technology of computing. All these phenomena appear in our daily life. Researches from biological sciences summarize their findings into a formative theory system, which benefit distributed computing in many areas, such as resource utilization, load balancing and also clustering. The innovative ideas proposed by Jim are not new in biological sciences, but it is an adventure in computing. So Jim spends some time with other professors in discussing the adventurous nature of this new idea. In this scenario, there is a group of people involved. One is Jim and the other is his co-workers. The knowledge units reside in Jim’s research ontology is not enough to solve the problem. So he communicates with people within and outside his domain. When domain is expanded into much more detailed areas, there might be some crossing area shared within the domains. When we define the ontology for these domains, the ontology shared between them can be reasoned in both directions. Also the discussion conducted between Jim and other professors should cover the resources referenced in the common ontology. From the KaM perspective, we can see that the domain information is crucial for KaM. It covers the first and the second requirements of knowledge advantage. Meanwhile it calls for information which needs to be corrected and also targeted. The ontology information and agents provide the basis for satisfying the third requirement. It should be effectively reaching the required information in a real time manner. The efficiency requirement calls for all resources easy to distinguish and categorize. Meanwhile this architecture should be applicable for all resources. Based on this feature, we bring in the idea of JAN which is an abstract object for all the general resources. It provides an abstract layer above the resources to achieve the uniqueness for different users share on the same resources. The JAN object is constructed according to the IEEE LOM (learning object metadata) standard. To better organizing JAN, we use the Resource-Oriented Architecture (ROA) architecture as the resource infrastructure for the whole model. The ROA architecture not only allows us to neglect the issue caused by resource duplication, it also eliminates the issues caused by resource control. All operations supported in our ROA implementation are stateless, which support our distributed architecture when the knowledge network data storage expands in an exponential speed. Also the ROA resource naming strategy provides us a more effective way in consistency checking and resource organization. From user perspective, they would leverage knowledge more effectively, which demands that the KaM be aware of user context. In our approach, we defined the context by analyzing user ontology pattern. The user ontology construction process is relying on calculating similarity between user's JAN and taxonomies refereed from universal ontology. The foundation for this calculation is based on the phrases extracted from both the sources. we developed our own key phrase extraction algorithm, the VKE algorithm [13] which combines the statistical information and heuristic rules together. Once user ontology is constructed, we applied two methods to detect the user context. First is the timeline model, according to the distribution of user activities along the timeline, we selected the taxonomy with the highest probability as user context. Second is to monitor user behavior and generate the transition model upon user's interest score. We predicted user's context based on the conditional probability of taxonomy, with which the user may be moving on.

3.1 Vijjana model

Vijjana is the detailed implementation of KaM concept. It works for one domain with ontology information and facilitates with agents running on it on the purpose of acquiring knowledge. The vijjana model is defined as:

Vijjana-X = {J, T, R, dA, oA, cA, vA, sA, rA, CA},

where X = the Domain name;

J= the collection of knowledge units called Jans in the Vijjana-X;

T = the Taxonomy used for classification of Jans [knowledge units];

R= the domain Specific Relations;

d_A = the Discovery Agent which find relevant Jans; o_A = the Organizing Agent which interlinks the Jans based on R;

c_A = the Consistency / Completeness Agent;

v_A = the Visualization Agent,

s_A = the Search Agent,

r_A = the Rating Agent,

C_A = the Collaboration Agent.

In the Vijjana model, the discovery agent (d_A) helps user to gain new knowledge units through performing key feature extraction and comparison between universal ontology. Organizing Agent (o_A) and Consistency Agent (c_A) are responsible for finding the right user taxonomy to place the knowledge unit. Visualization Agent (v_A) construct the knowledge network from a users personal perspective or any global views and display them in several structured format such as radial view. Based on all of the above, we will present the KaM architecture with the Figure 1.

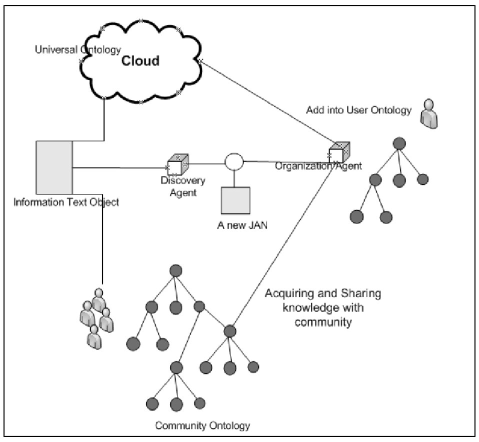

Figure 1: KaM Architecture

4. Our Implementation

4.1 Implementation: The hierarchy structure

Being different from other resources, knowledge usually is regarded as an abstract object. To better interpret that, people classify knowledge into different categories, which enrich knowledge unit with a hierarchy of information. Some resource websites, like open directory [11], already provides users a large amount of knowledge units with well classified ontology information hierarchy. This Meta information is based on the assumption of Resource-Oriented Architecture (ROA) [12] and also addressing a simple resource retrieval problem. Also, according to the Web 3.0 standard, knowledge unit as a type of information, should be retrieved with semantic relation, which to some extent, is extracted from information hierarchy.

4.2 Construct: User knowledge abstraction layer

The basic knowledge unit in the Vijjana, JAN, is abstracted from one of the text objects, like document, webpage, email and other information into a resource that can be reusable in the ROA architecture. By the definition of LOM, it contains important fields such as, annotations, references and also categories. These segmented information are helpful when dealing with knowledge alteration or recommendation. However this information should not be identical from different user's perspective. The annotations should not be simply a key word list generated by our Vijjana Key phrase extraction (VKE) algorithm; the references are also not a static list referencing other JANs; and moreover, one JAN may be mapped to more than more categories. Generally, a JAN is a representation of knowledge unit from a user. It contains meta information describing its resource, a conceptual meaning which is defined and adjusted by the user. In another words, a JAN is only meaningful when it is combined with user ontology information. So we need to have a user knowledge abstraction layer, which consists of JANs, and which are unique for different users

4.3 User profile based on ontology

Practically, a user always crosses over multiple domains. For instance, Professor Jim smith (as in section 3) has his domain in academic study. Besides this, he also has domains in personal life and others categories. Based on the KaM definition, one KaM resides only in one domain. For a particular domain, it owns ontology which defines its resource classification and relationship among them. Here the ontology information is not only the generalized classification cited from universal ontology, also is enriched with personal preference. Considering this, a user profile augmented with user ontology is necessary. A typical user profile contains two parts. One is the user's preferences and the other is the user's behavior rules. The prior part is mostly the user ontology, generalized from user's behavior history that records which web page the user browsed or what document the user read. From the content of the information object, an agent can generalize the classification and converge them into user preferences.

When a new information object comes in, the O_A in Vijjana framework will find a suitable category for it to reside in. This process is transformed into queries upon user's profile and the return result reflects the category information. In Vijjana model, the T set defines these concepts as taxonomies. For each JAN, its original resource content is extracted using VKE and a similarity test is performed along all user taxonomies to determine which is the most matched concept. If there is no concept in, which requires the probability should be over the threshold probability, then a consecutive similarity test is performed on the universal ontology. Based on [15], approximately 3,000 terms will cover all general concepts for a specific domain. A user, using finite taxonomies can cover all domains he/she crosses over.

4.3.1 Construct: User ontology

To construct user profile with ontology information, we used the Open Directory Project (ODP) as our referenced ontology. The ODP is regarded as the largest taxonomy store for web directory. The taxonomies are organized in hierarchy structure. In [12], authors concluded that only the first three levels of the taxonomy as references will promote the ontology hit accuracy. In Vijjana framework, we would also use the taxonomies in the first three levels as our concept set. For a taxonomy used in the universal ontology, we will train it with a collection of documents. For each trainable document, we preprocess it by stemming algorithm and extract phrases form it using our VKE algorithm. And then merge all key phrases into one vector in which each key phrase is distinct. In our VKE algorithm, phrases are evaluated by its entropy value. Here we used the phrase entropy value as its weight. When a new JAN is brought into vision, its reference document also needs to go through the stemming process and the key phrase extraction process. Once we have these two vectors, new vectors are generated from JAN's document and taxonomy vectors, and we need to apply the cosine similarity method to determine which taxonomy the JAN should reside in. Another issue of the method is: it loses the user preference since its weight is same as one in the taxonomy. Considering from user perspective, the keyword of JAN should be verified by user to strengthen the view of the user. Before we apply the cosine similarity, there is a normalization process to ensure the weight value is between 0 and 1. This step ensures that the final cosine value is between -1 to 1, where its absolute value closer to 1 stands for a higher similarity. For each word in both vectors, its weight w_i is calculated using following formula:

$$w_i = \frac{w_i}{\sqrt{\sum_{i=0}^{n} w_i^2}} \quad (1)$$

The cosine similarity of vectors can be expressed as:

$$cosine(t_i, jan_j) = \sum w_{ik} * w_{jk} \quad (2)$$

where, w_{ik} indicates the weight of term k appears in the taxonomy i and w_{jk} indicates the weight of term k appears in the new JAN j. For term that doesn't appear in either vector, the weight should be 0. The highest similarity taxonomy should be selected as part of user ontology.

4.3.2 User behavior model

Another part of user profile is the user's behavior model. For a finite number of taxonomy, we can construct finite rules to describe the transition between user activities. From the top level ontology, the rules are more generalized compared to detailed rules derived from the lower level. Also the next state in transition can also be predicted using a statistical probability model. For instance, for a user who has a strong interest in sports may more likely to spend more time in reading sports news rather than financial news after finishing his daily chores. So the transition on switching from work to news is more likely to choose the sports news as the end state. This behavior model is used in context awareness analysis. In order to evaluate the interest, we keep an interest score taxonomy hit number as its interest score. For each taxonomy, the interest score is calculated with

$$I_{t_i} = \frac{total\_hit\_number}{taxonomy\_size}, Where \qquad (3)$$

the I_{ti} stands for the taxonomy i, and taxonomy size is the number of JANs in this taxonomy. Using the Interest score we can construct the relationship between taxonomies. This is explained in the following section.

4.4 Context awareness

As illustrated above, a user behavior model is the base for the context awareness. To detect which context the user resides in, the basic method can fall into two categories: timeline and knowledge hits statistics. Before explaining our context awareness method, we need first to define what is a context. From its semantic meaning, a context should stand for: *where is the user*. In the KaM concept, we can define the context as the ontology the user resides in so that the context is defined with a main taxonomy as the current state and several sub-taxonomy as the possible next state. Context = {Current, {Next}}.

4.4.1 Timeline context awareness

In the Vijjana model, a JAN is contributed by the user. For each one, Vijjana records it with a timestamp as its submission time and revision time. On daily basis, for a particular user, if in certain time period there is a distinguishable increase of JAN submission or revision for certain taxonomy, then we can mark this time period with this taxonomy and correspondingly pick it as user's context for this time slot.

Using this method, one day is divided into two periods, user active period and inactive period. The basic time unit can be set as one hour. The inactive period is time units without any user activity. In opposite, user has activities during the active period. For a certain long enough time phase like one week or month, the user activity can be categorized by these two periods. For the user active period, we can cluster the taxonomy if we already know taxonomies the user owns. Based on the user activities for taxonomies, we can calculate the probability for each taxonomy on time phases, P ($t_i|time_j$), and then choose the highest one as the user's context. We can see this from the Figure 2

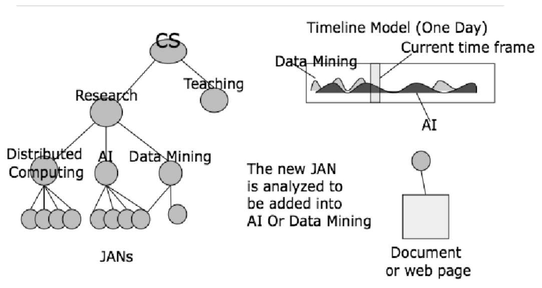

Figure 2: The Time Line model

4.4.2 Interest Driven Context Awareness

Another context awareness model has concerns on the taxonomy interest I_{ti} . For each concept, it has an interest score calculated. For all concepts, statistically each concept has factorial a value which is between zero to one to stand to support for its hit probability.

$$PI_{t_i} = \frac{I_{t_i}}{\sum I_{t_i}} \qquad (4)$$

By this interest score, we also can calculate the probability of certain transition to happen, which is a conditional probability of user moving from one taxonomy to another one, P ($t_{next}|t_{prior}$). The taxonomy with the largest probability stands for (more likely) user may be moving on, as the above example illustrated in user behavior model. According to the conditional probability, we formed a priority queue which stores the possible taxonomies ranked by their probabilities. The next state of context is selected from this queue. Meanwhile a *happened* transition will update the interest score and consequentially update PI_{ti}. Generally, for a small knowledge network with low average hit number, the update operation would not be costly. However when treating with knowledge network, the update should be done only when reaching hit percentage threshold, like 5%, which controls the update frequency.

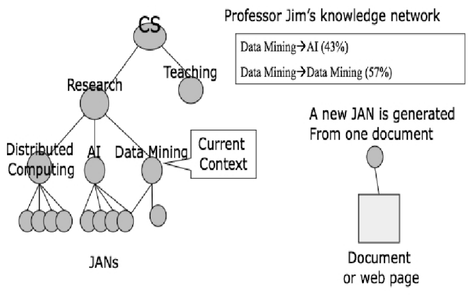

Figure 3: Interest driven model

4.5 Knowledge discovery process

One user, at any time context point, should reside in only one context. As defined before, the context provides the current taxonomy and next possible set of taxonomies. For each user, his/her ontology shares part of the universal ontology. By 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 on computer science. For the communities, all share the universal ontology. This also leads to our knowledge discovery process into a three-step procedure. We call this procedure set as "*Call it once*". The discovery first happens locally in the user context, then expands to communities where the user resides in the same ontology, and then finally explores the universal cloud. The knowledge discovery can be initiated by a user in certain context, or by an agent during context switching. No matter in which way it goes, it is performed by queries upon taxonomy. A typical query is constituted by a set of phrases.

4.5.1 Knowledge discovery: Local

The local search is confined in user taxonomies. Within one taxonomy range, we can use phrase matching on taxonomy vocabulary. If it appears in the taxonomy vocabulary (keyword set), all JANs related to this keyword are returned, ordered by its weight. Otherwise, we need to look up its synonyms and process the searching procedure again.

4.5.2 Knowledge discovery: Community

Since people in the same community share the same ontology, so we can use the collaborative filtering (CF) technique to recommend JANs to users. Recall in the KaM model, in order to eliminate the problems for organizing and consistence checking, we applied the ROA architecture which required all items should be uniquely identified. For an original resource, it is abstracted in to JAN to add into user's knowledge network, which is unique in the whole knowledge network. So here we can't directly apply the CF technology upon the JANs. There are two methods for solving this problem. First, also a rudimentary one- is to use the JAN's referencing the original resource as item. The other is using 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 is the hit number of the JAN. Here it is:

$$UI_{ij} = \begin{cases} hitnumber_i & \text{if item i appears} \\ & \text{in } user_j\text{'s KN} \\ 0 & \text{Otherwise} \end{cases} \qquad (5)$$

Once we have this matrix, we can use the adjusted cosine similarity to compare two JANs. The JANs with highest similarity should catch the user's eye. For the second method, we use the keyword to replace the JAN, so the user-item matrix is formed as in the following expression. Here the keyword stands for a set of JANs which use this keyword as index.

$$UI_{ij} = \begin{cases} 1 & \text{if keyword i appears} \\ & \text{in } user_j\text{'s KN} \\ 0 & \text{Otherwise} \end{cases} \qquad (6)$$

We calculated the similarity between the two key-words and recommend were JANs indexed by the highest similarity keyword for the user.

4.5.3 Knowledge discovery: Universal

Since the second step of "call it once" has happened on community ontology which is also a part of the universal one, the final step is to search along with ontology which the community ontology relates to. These relationships are defined in the universal ontology and the search is already out of user's context. So here the query is without any user preference. For each related ontology, we performed a local search on all its taxonomies and returned JANs as a compensation of result from local search and community search. The overall discovery process can be viewed as the Figure. 4

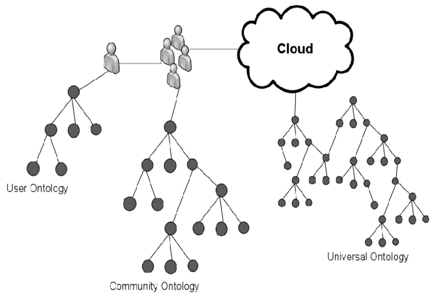

Figure 4: The whole discovery process

5 Conclusion

In this paper, we discussed the knowledge discovery problem coming with the current enormity of information overhead we encounter in web searching. To address this problem, we proposed the idea of the KaM, which is a new architecture designed to help people to effectively retrieve knowledge with ease and contextual support from peers. Inside the KaM, we created user ontology by using two well defined user context awareness models, one is a "timeline" model and the other is the "user interest driven" model. And then, we applied a novel knowledge discovery approach: "Call it once" to retrieve any desired knowledge. This approach transformed the discovery sequences into three basic steps: local, community and universal. The whole architecture is founded upon the ROA architecture, which ensured a high utilization of available resources.

6. References

[1] T. R. Gruber, Toward Principles for the Design of Ontologies Used for Knowledge Sharing. In: International Journal Human-Computer Studies. Vol 43, p. 907-928.

[2] H. Chen and T. Ng, An algorithmic approach to concept exploration in a large knowledge network (automatic thesaurus consultation): symbolic branch-and-bound search vs. connectionist Hopfield net activation," Journal of the American Society for Information Science, vol. 46, pp. 348-369, 1995.

[3] http://www.deepamehta.de/.

[4] L. Sauermann, G. Aastr, M. Kiesel, H. Maus, D. Heim, D. Nadeem, B. Horak, and A. Dengel, A.: Semantic desktop 2.0: The Gnowsis Experience, in International Semantic Web Conference. Volume 4273 of Lecture Notes in Computer Science. Springer, 2006, pp. 887- 900.

[5] W. Woerndl and G. Groh, A social item filtering approach for a mobile semantic desktop application," 2008.

[6] H. Lieberman, Letizia: An agent that assists web browsing," in IJCAI, 1995, pp. 924-929.

[7] D. Mladenic, Text-learning and related intelligent agents: A survey," IEEE Intelligent Systems, vol. 14, no. 4, pp. 44-54, 1999.

[8] S. E. Middleton, N. R. Shadbolt, and D. C. De Roure, Capturing interest through inference and visualization: ontological user profiling in recommender systems," in K-CAP '03: New York, NY, 2003, pp 62-29

[9] C. C. Chen, M. C. Chen, and Y. Sun, Pva: A self-adaptive personal view agent, 2002.

[10] H. R. Kim and P. K. Chan, Learning implicit user interest hierarchy for context in personalization," in IUI '03: Proceedings of the 8th international conference on Intelligent user interfaces. New York, NY, USA: ACM, 2003, pp. 101-108.

[11] O. D. Project, http://dmoz.org," April 2002.

[12] L. Chen and K. Sycara, Webmate: a personal agent for browsing and searching," in AGENTS '98:Proceedings of the second international conference on Autonomous agents. New York, NY, USA: ACM, 1998, pp. 132-139.

[13] L. Wang, Vijjana key phrase generating algorithm, *unpublished: WVU internal document*.

[14] L. Richardson and S. Ruby, Restful Web Services. O'Reilly Media, 2007.

[15] H.M. Haav and T.L. Lubi, A survey of concept-based information retrieval tools on the web," in 5th E.European Conference, ADBIS 2001, Vilnius, Lithuania, 2001.