A Context Centric Model for building a Knowledge advantage Machine Based on Personal Ontology Patterns

Luyi Wang
West Virginia University

A Context Centric Model for building a Knowledge advantage Machine Based on Personal Ontology Patterns

by

Luyi Wang

Thesis submitted to the
College of Engineering and Mineral Resources
at West Virginia University
in partial fulfillment of the requirements
for the degree of

Doctor of Philosophy of Computer Science
in
Computer Science

Lane Department of Computer Science and Electrical Engineering

Morgantown, West Virginia
2012

Keywords: Context Centric, Ontology, Ontology Pattern, Knowledge Advantage Machine, Artificial intelligence, Information Retrieval,Software Engineering

Abstract

A Context Centric Model for building
a Knowledge advantage Machine
Based on
Personal Ontology Patterns

Luyi Wang

Throughout the industrial era societal advancement could be attributed in large part to introduction a plethora of electromechanical machines all of which exploited a key concept known as Mechanical Advantage. In the post-industrial era exploitation of knowledge is emerging as the key enabler for societal advancement. With the advent of the Internet and the Web, while there is no dearth of knowledge, what is lacking is an efficient and practical mechanism for organizing knowledge and presenting it in a comprehensible form appropriate for every context. This is the fundamental problem addressed by my dissertation.

We begin by proposing a novel architecture for creating a Knowledge Advantage Machine (KaM), one which enables a knowledge worker to bring to bear a larger amount of knowledge to solve a problem in a shorter time. This is analogous to an electromechanical machine that enables an industrial worker to bring to bear a large amount of power to perform a task thus improving worker productivity. This work is based on the premise that while a universal KaM is beyond the realm of possibility, a KaM specific to a particular type of knowledge worker is realizable because of the limited scope of his/her personal ontology used to organize all relevant knowledge objects.

The proposed architecture is based on a "society of intelligent agents" which collaboratively discover, markup, and organize relevant knowledge objects into a semantic knowledge network on a continuing basis. This in-turn is exploited by another agent known as the Context Agent which determines the current context of the knowledge worker and makes available in a suitable form the relevant portion of the semantic network. In this dissertation we demonstrate the viability and extensibility of this architecture by building a prototype KaM for one type of knowledge worker such as a professor.

Acknowledgements

In the past five years I spent in West Virginia University, all merits came with supports and inspiration from all people around me. I would first like to thank my committee chair and advisor, Dr. Ramana Reddy. Serving in a position for inspiring and providing guidance on my research interest, He is far beyond this scope in influencing me with always positive attitude and providing endless support. What I learned from him laid a solid base for my long term career and also is a fortune for my future life. It is indeed my honor to be his student.

I would also like to thank Dr. Sumitra Reddy, Dr. James Mooney, Dr. Arun Ross and Dr. Asesh Das for being on my committee. Without their help and guidance on my research, I can't finish this topic with the current achievements. Specially thanks to Dr. Arun Ross and Dr. Asesh Das, every discussion we took on my research topic inspired and benefited me to think it in a more considerate and deeper way. All these drove this topic further and further.

I would also like to thank students in CERC lab with whom I've had the pleasure of working alongside. Because of them, this research topic was developed into a well-rounded shape. I would also like to express my gratitude to my friends. Specially to Dr. YunFei Zheng,Trevor Kemp and Yutian Gan. All of them gave me warmhearted help and support, which was a driving force pushing me forward till this step. Special thanks to many friends, like Yu Feng, Yaohui Ding, Wu Nan, Shan Yang, Lin Chang and all other friends whoever helped me. All of them companied me along with the tough road. We fight we smile.

Finally I would like to thank to my parents. Thanks for their love on me that always be my momentum for reaching and achieving a higher goal. They guide me to understand there are many importances in life and their encouragement is always along with me, bringing me endless power to accept challenges.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 History

Has the Internet changed the method by which people acquire knowledge? To answer this question, we need to look back in retrospect to the time when people would use a brick-and-mortar library to find reading material for their subjects of interest. In a library, resources were limited. Not only were knowledge materials for a given subject limited but so also was quick access to colleagues and friends that shared an interest in the subject. Additionally, placing ourselves in this antiquated reality reminds that discussions within a group of colleagues usually consumed large quantities of time and had a large financial burden due to travel and communications technology costs. Regarding these points, we can give demonstrable examples in the present day in search technology like Google and social media like Facebook. These tools have given us starkly different methods for disseminating information than what came before them. However, have these technologies truly changed the way we acquire knowledge?

An easily consumable list of knowledge on a topic is not directly provided by search engine technology that simply consolidates all information related on a specific set of keywords. For example, searching on keyword "apple" will return resources that are predominantly about Apple Inc. On the first page, minimizing matches that contain knowledge on the fruit we eat everyday. The search results benefit people who are interested in knowledge of Apple Inc. but the results are not suitable for people who try to know more about the fruit. Also, much irrelevant information accumulates alongside useful information when people generate content by repeatedly discussing many aspects of a topic. This causes difficulty for people who search for information on one specific aspect of a topic. It is hard for them to filter out conversations irrelevant to their interest. For example, when people discuss the topic "cell phone" the discussions may concern phone hardware. These discussions range in specificity from phones' technical specifications to their outward appearance. Content authors may discuss topics like a phone's running platform, particular functionality of the device, or a service carrier for it. They may also refer to its manufacturer, perhaps discussing the history or current trends of the company. For different groups of people, the way of distilling search results into usable information varies. An engineer who intends to learn smart phone programming expects different knowledge compared to a prospective customer who would like to purchase a new phone. All these attest to the fact that the knowledge discovery process from an individual's perspective is not a straightforward process.

1.2 Methods Applied

In regard to these problems, researchers in Artificial Intelligence have invented many rule-based expert systems [**?**][**?**][**?**][**?**]to infer information from large amounts of incoming data. Discovering knowledge in databases by way of data mining focuses on retrieving useful pieces of information in a large volume of data to make optimized decisions [**?**] by utilizing neural networks[**?**][**?**], Bayesian learning[**?**], and other statistical methods[**?**][**?**]. From an algorithmic perspective, researchers are putting effort into developing better algorithms[**?**][**?**] to eliminate less influential features. Meanwhile, with the development of communication technology, pervasive computing has emerged as a new area in the applied field of information retrieval. Sensors such as GPS modules, cameras, and accelerometers are built small enough to be embedded into portable devices[**?**]. These portable sensors make it possible for information to be obtained in real time and also bring in a new challenge that knowledge must be presented on demand. Consequentially, the methods for retrieving this information must be revised as well. They are not big different from the original methods that call for heavy computing power in discovering useful information from a large volume of stored data but bring in new requirements for efficiency and correctness.

1.3 Context Centric approach

Rule based systems like the ones mentioned aim to solve problems by discovering an optimal solution or discover rules for making better decisions. When search engine technology opens the door for ordinary people to search vast knowledge repositories on the Internet and when sifting through the content in the information boom becomes an affair in which end users must take part, it is required to combine mining techniques with information about a user,which provides a context and largely reduce the applicable scope of knowledge . The traditional approach for accomplishing this is to build up user profile by monitoring, accumulating and analysing a user's behavior. A user-monitoring agent, running on desktop computers, records user's activities such as web browsing history, file operations, etc. Based on the duration and frequency of predefined types of the user's activity, the agent can incrementally learn and create a user profile that automatically determines a user's habits and preferences. When information collection becomes a real time manner, query time efficiency becomes more important than ever before, and relying on user preferences to aid in data mining is not enough. The degree of correctness or accuracy of search results also turns out to be measured by the user's interest. This user is the final judge in determining if the queried knowledge is useful. Moreover, the measurement of result accuracy is largely increased by an awareness of context that reflects the user's circumstance. This changes the knowledge discovery process to be a user centric model, which think in the user's view and learn in a specific context. And so well the discovery process needs to be intelligent in acquiring user context and also be capable of switching when the circumstance changes. For instance, when a person who has a dinner appointment in a different city queries restaurant, he/she is more likely to find the best city in that city rather than around.

If we think in a more theoretical way, all user behaviors are observations and user habits are transitions between user preferences, which act as states of the user's profile. On a certain state, knowledge should be filtered according to the corresponding preference. Meanwhile, a user habit transitions state to another preference and therefore knowledge should be filtered again. For a particular user, his/her preferences should be covered by finite abstract concepts and the habits can also be enumerated.

We can use an example to illustrate this. Prof. Smith checks his email everyday when he wakes up. He usually goes through his student's email first, then colleagues and others. This habits can generalized as

$$S_{wakeup} \rightarrow S_{checkemail} \tag{1.1}$$

$$S_{email} := S_{Student} + S_{Colleagues} + S_{Others} \tag{1.2}$$

Here, we use S to denote states. There are two transitions, from "wakeup" to "check email" and once in "check email" from students to colleagues and others. They share a commonness that they are all Prof. Smith's habit but also different on causal reasoning. From wake up to check email is a time line change. It is a habit upon time. The order in checking emails is a user habit upon preferences.

From this view, we need to build user context models in order to better describe user preferences. The user preferences can no longer to be static and isolated states but are connected by user habits which act as transitions between the preferences.

1.4 Knowledge engineering on user preferences

As we mentioned, user preferences should be considered as a finite set and described by enumerated observations on user the behaviour model. Each of these observations need to be engineered into knowledge units which map to a certain user preference.

While working on this project, we noticed an interesting phenomenon. A knowledge unit usually doesn't fit well into one particular preference. Usually, a knowledge unit itself can be viewed from several perspectives. For example, let us imagine that Prof. Smith received an email from students inquiring about a course assignment. This email contains plenty of information covering multiple areas like Prof. Smith's email, student's email, and course assignment. Theoretically, we can formulate this scenario like so: for this email, because it is sent by a student to Prof. Smith, it connects two concepts: student and professor. Meanwhile it is about one assignment in one course. It connects one more concepts: assignment and course. We can use the following formula denoting these relationships. Here we use j denoting a knowledge unit and C denoting a concept.

$$j(email|Prof.Smith) \rightarrow j(email|student) = C(Prof) \rightarrow C(student) \tag{1.3}$$

$$j(assignment_inquery) \rightarrow j(course) = C(assignment) \rightarrow C(course) \tag{1.4}$$

We also notice that it is about Prof. Smith's email, so the context is email and user is Prof. Smith. However concerning the relationship between the preferences, we utilize an ontological concept that was originally designed for explaining domain knowledge. A ontology is regarded as a speculation of concepts and the relationships among them [**?**]. From its definition, we can tell that ontology defines the relationship between concepts in a more generalized way than compared to habits, which only concern behaviour rules as pertaining to particular user. However the concepts and habits share commonality in that when certain habits are applied, concepts covering them are also applied.

In the previous example, Prof. Smith may transit from C_{email} to $C_{assignment}$ to figure out what's the student's question about. Then this email, considered as a knowledge unit, connects two concepts,email and assignment, together because it is a email about assignment. More generally, Prof. Smith may receive many this type of student email everyday. To better organize his email, he usually categorizes them into a student-assignment group. So here there should be a relationship established between these two concepts. This relationship should be applicable for all professors like Prof. Smith who teach courses. If Prof. Smith has a personal ontology defining all his knowledge and so well are all other professors, then from this we can tell the partial ontology concerning student-assignment-email should be similar among all ontology of professors. From this perspective, we can tell that a user ontology should conform to an entire group one in which individuals share the same interest but in addition should have personal versatility. From a knowledge engineering perspective, a knowledge unit should be capable of mapping to different concepts with enriched meta-information in distinguishing the difference.For example, a smart phone should be tagged with OS information in developer view but manufacture information in a consumer view.

Consequently, concepts share knowledge units when they have connection defining by user indicating their relationship.

1.5 Problem we need to solve

As discussed already, we apply ontology in illustrating the user preferences and their relationships. Because of the challenges in time efficiency in real time information retrieval, there is a requirement of the context model that details a user's circumstance and reflects the current ontology the user is working with. Moreover, the information retrieval and collection process becomes more systemic in distilling information to map into a user's personal ontology. The user ontology in some degree is similar to a group ontology which consist of people who own the same interest on certain knowledge set. This brings up the requirement that a context model can communicate with other ontologies to retrieve common interest information.

1.6 My Contribution

In my dissertation, I propose a new architecture that is suitable for building up a context centric knowledge network. The architecture is named as Knowledge Advantage Machine (KAM).The knowledge advantage stresses the importance of knowledge and the machine is concerned with the architecture. My main contributions I did to build KAM are mainly in the following parts.

- Three tiers infrastructure
- VKE algorithm
- Context Model

The novelty of this architecture is that we build up a three tiers architecture that personal ontology, community ontology and global ontology are all connected together. The personal ontology is created by user but mapped with global ontologies. In a particular ontology, all the knowledge units, JANs, are processed using my own keyphrases generation algorithm and organized into a user defined logic layout. To reveal this layout, I apply the RESTful framework that allows us to better utilize resources and easily retrieve them. In order to reflect the user preferences, I design two context model, timeline model and interest driven model. We use these two models to detect user current taxonomy and correspondingly provides user the related knowledge units, JANs, using our query method “Call it Once”. The “Call it Once” query is different with traditional wild search that it is performed on these three tiers. The local search will return user’s personal taxonomies and JANs, the community search will return knowledge units from people who has the same taxonomy with you. The global search is a wild search that return the related global taxonomies.

In this Chapter, I already emphasis the knowledge importance, the current situation and the problem we are solving. The second chapter focuses on the previous work done in semantic work, ontology and user profiling. Chapter three illustrates the concept of KAM and whole architecture. Chapter four pertains to the architectural design, which is an implementation of the Restful structure. And the key phrase extraction algorithm VKE is illustrated in Chapter five. In Chapter six, I explain the methods I apply in building up context centric patterns utilizing user ontology information and introduce a workflow that we have termed “call it once” that divides the whole information discovery process into three phrases. In Chapter seven, I reveal the real KAM implementation and Chapter eight is a conclusion chapter that concludes all my contributions.

Chapter 2

Related Research

In this chapter, I briefly review related research in knowledge engineering field. It covers the topics of knowledge engineering, Ontology, semantic Web service and context awareness technologies.

2.1 Knowledge Engineering development history

The concept of knowledge engineering was proposed by Edward Feigenbaum and Pamela McCorduck in 1983 in book "Fifth generation"[?]. The book introduced a "new" generation of super computer designed in Japan that provides tremendous parallel computing power to support development in Artificial Intelligence(AI). It was for the first time proposed that knowledge need to be processed and represented in various types. Computer technologies in processing large scale data base [?]and knowledge inferences [?]became popular in 1980s. These ideas were soon incorporated into the development of expert systems which, by then were usually rule-based systems conducted by interviewing experts in fields, and implemented for specific purposes. As a consequence, their knowledge was tightly tied with human defined rules. As time went on, people[?] realized that rule-based systems disobeyed the principle of knowledge reuse for their lack of knowledge representation. Meanwhile, on the cost consideration, as a ruled-based system expanded into a larger scope, the system became harder to maintain for the user spends more efforts in remembering and maintaining the reasoning logic amongst a huge amount of complex rules. Therefore, since the late 80s computer modelling [?] with an expandable and reusable knowledge base became a new trend. Studer[?] reviewed in 1998 the two phase development of Knowledge-based system in the 80s and also shared insights onto a new trend of the 90s, the development of ontology. The popularity of ontology was attributed to its promise on easy interpretation between human and machines across domains[?]. The terminology was borrowed from the philosophical concept of ontology that describes the existence of objects. In knowledge engineering, Ontology denotes a knowledge representation with selected properties that allow automated processing, formalizing and reasoning in certain domains.

2.2 Research upon ontology and semantic web

McCarthy first introduced "ontology" to computer science in 1980[?]. The concept became popular after 1993 when Gruber[?]proposed his principle in developing ontology and conducted relevant case studies. In Gruber's paper, it was explained that ontology is "an explicit specification of a conceptualization", and that ontology could be used as agreements on knowledge sharing in building reusable knowledge components and knowledge-based web services. Gruber's proposal were well accepted in the knowledge modelling society. Guarino and Giaretta in 1995 [?] redefined the Ontology terminology in computer science to avoid confusion.

In contemporary knowledge engineering, the core problem still lies in the searching of commonality within a large of amount data. Even though a growing number of knowledge services are provided as means for searching (search engine) or referencing (information site, e.g., Wikipedia), the concept of reusable knowledge is still to be defined. Along with theoretical development, researchers also emphasized algorithm design on data mining techniques. In 1995, H. Chen and T.NG [?] proposed two knowledge discovering algorithms. The first algorithms utilized a semantic network on structured knowledge to traverse along the large scale knowledge network to explore related conceptual knowledge. The second algorithm relied on the parallel relaxation of neural networks. Both methods quantified the knowledge relationship as a measurement of their similarity. The structured knowledge representation grants efficient information retrieval and provides the basics for information reference. With the emergence of OWL standard, structural information reasoning and inference gains more interest in the field, too.

In [?], Deborah and Paulo (2004) suggested a new web structure, inference web, which makes inference proof along reasoning with a series of tools. They demonstrated the concept with a knowledge based registry method, an inference engine, and utilities for knowledge representation. Meanwhile, the semantic web services are also proved to be suitable for new infrastructure [?][?]. In[?] , LEO and his colleagues discussed their experience on building a personal semantic desktop system(2006). The discussion was mainly focused on ways to organize the resource for personal users by meta-data extraction. The knowledge sharing aspect of the system was confined within wiki page authoring. In 2008, the semantic web was brought to mobile networks[?].

2.3 Research upon user profile and context awareness

Along with the academia and industrial efforts in knowledge discovery and semantic reasoning, another trend in knowledge engineering focuses in personal knowledge networks and the correlated contextual information. One relevant field is the construction of user profiles. In 1995,[?] proposed to build user profiles by monitoring the web-browsing history of the users. In 1997, Thomas introduced a method of query analysis upon the bookmarks[?]. A user profile, however, is not limited within web pages. Efforts were also made to expand the profile range from web pages to a wide range of information sources, such as documents and emails[?][?]. Since 1999, the construction of user profiles started to utilize techniques to categorize information with abstract structures. In [?], [?], it was proposed to enumerate lists of user's interest. [?] introduced a method to organize information into hierarchical concepts. Another relevant field of study is the context awareness. The initial context awareness research is achieved through questionnaires, which confines user context based on the user's feedback. In order to better representing the user context, Researches [?],[?] categorize user interest into classes and form hierarchy structure of these classes. Joana [?] in 2004 introduced a method that relies on ontology to define the user context with the help of the Open Directory Project[?]. Most research defining user context is based on the vector space model[?] that calculate with the tf*idf method the weight or rank of each query phrase. The context is selected from the list of the top-ranking user concepts. In 2004, Middleton[?] used the K-nearest neighbor method to build user profiles, in which the user profile construction began from a cold start. And the accuracy of finding user preference is improved with accumulative adding new information. Joana [?] verified the idea by testing the convergence of profiles, its ranking orders, and the pruning non-relevant concepts. For user profiles, it was found that with merely half of the ontology information a user profile can converges into a stable state. For the ranking orders, no significant difference was reported between representation based ranking and VSM weight based ranking. For the pruning, it was discovered that the process improved the accuracy in context definitions.

Chapter 3

KAM Architecture

In this chapter, I first introduce of "knowledge advantage machine" model and illustrate the KAM (old Vijjana architecture [?]) we built for this concept. All these work are related to an ongoing project in our SIPLab of West Virginia University,which is aiming at building up a real Knowledge advantage machine that can help user organize their personal knowledge network.

3.1 Knowledge Advantage Machine Background

Throughout the industrial era societal advancement could be attributed in large part to introduction a plethora of electromechanical machines all of which exploited a key concept known as Mechanical Advantage. In the post industrial era exploitation of knowledge is emerging as the key enabler for societal advancement. With the advent of the Internet and the Web, while there is no dearth of knowledge, what is lacking is an efficient and practical mechanism for organizing knowledge and presenting it in a comprehensible form appropriate for every context. This is the fundamental problem addressed by Knowledge advantage Machine.

Our purpose of building a Knowledge Advantage Machine is to help user exploiting a large amount of knowledge to solve problem in a shorter time. The knowledge in use is selected from user ontology in which knowledge units are organized in different domain scopes where the user applies knowledge across over. The knowledge unit is instances defined by user with enriching meta information. This building KAM principle lies in an assumption that a user knowledge should be limited in a certain scopes instead of a universal realm.

3.2 KAM Features

Before we explains the KAM architecture, we need to first examine the areas the KAM should apply on. Here we use several scenarios to illustrate the problems KAM should cope with.

3.2.1 scenario One: Domain information

Dr. Watson is an assistant professor in CSEE Department of West Virginia University. His major interest is in Artificial Intelligence. He usually works more than ten hours a day, teaching courses and doing research. He teaches two courses with around seventy students in total every semester. Meanwhile, he has his own AI research lab. There are four master students and three PhD students working under him. His whole family lives in Morgantown,West Virginia. He has a beautiful wife and a five years old boy.

So in this scenario, Watson plays two roles in his daily life. One lies in academic domain and the other lies in family.

- Work Role: a Professor who conducts research and also teaches courses
- Family Role: A husband has wife and child.

Based on this, we can see the domain information varies with people's role changes. When Dr. watson focus on work, He resides in his academic domain. When he come to home, the role changes to a family member, correspondingly the domain alters to be family. From this perspective, KAMs used by Dr. Watson should also reside in two domains. If we define a KAM in each domain, then these KAM should switch their leading role when context changes.

3.2.2 Scenario two: Communication within Domains

Dr. Watson teaches undergraduate Artificial Intelligent class every Monday, Wednesday, and Friday. In the beginning of his lecture, he usually gives out all lecture references for the giving lecture which he collected with time goes on. Even though the topic every year is same but he is used to add some new materials in his lecture, such as new technologies trend, to attract student's attention. On a particular night before the lecture, after reading some new papers published by peers in another university, Dr. Watson is caught by one paper illustrating the new study on a course planning system. This paper illustrates an new experiment of applying a revision of old COCOMO cost model in effectively planning courses. In order to let student have a better understanding on the COCOMO model, he plans to leave a paper review upon this paper as a part of new assignment.

From this scenario, we can tell that, in a more abstract way, Dr.Watson already constructed his own knowledge network in a mental way. He kept updating his knowledge network by synchronized with works from others who are in the same filed. However the current way of synchronization was manually done by Dr. Watson himself. He manually found papers to read and determined if it was worth to save in his mental knowledge network. From this perspective, the KAM working in Dr. Watson's academic domains should be able to automate this process by communicating with peers in the same fields and keep Dr. Watson updated. If there was another KAM helping in this matter for his peer, it requires that KAM within same domain for different user should have the capacity of communication. These KAM should also share knowledge unit because of their commonness between domain information defined in user ontology.

3.2.3 Scenario three: Communication within same Domain but different area

On Tuesday Dr.Watson spent most of his time in doing research. He is writing a research proposal to apply one science funding. This would be a new project collaboratively cross several universities. He takes the lead role in communicating with professors in other *universities. They spend a plenty of time in discussing innovating ideas that utilize theories abstracted from biological phenomena into distributed computing. All these phenomena appear in our daily life. Researches in the biological summarize their finding into a theory system, which benefit distributed computing in many areas, such as resource utilization, load balancing and also clustering. The innovating ideas proposed by Dr.Watson is not new in biological but is an adventure in distributed computing. So he spends some time with other professors in discussing his new idea.*

In this scenario, more people from different domains involve. Dr.Watson's knowledge is in a limited scope that it is not sufficient to support in exploring his new idea. He consults help from other people who has the knowledge. Thereafter his knowledge network expands into a new area that has connection to his original domain. However his knowledge network in this new area is sparse. To construct it, he ask knowledge share from people who already have a perplex one on it. From this perspective, for the KAM residing in Dr.Watson's academic domain,because of the new expansion of Dr. Watson's knowledge network, KAM should construct a new taxonomy in its ontology.Also it should have the ability of getting knowledge units from the same taxonomy in peers ontology. That requires the KAM should be able to import other people's knowledge network. If the peer also has a KAM,when a new taxonomy appears in one KAM's ontology, the two KAM should communicates and share knowledge. If the knowledge network become too large to handle for one KAM, it should be able to split into multiple one. Each takes care one particular area.

3.2.4 KAM Features on domain

After on all above explanations, we can conclude that a KAM should contain the following features in domain perspective.

- One KAM should be defined within a particular domain, enriching itself with domain information. On one domain, it can contain more than one KAM.
- One KAM should reside in only one domain, but it can interactively communicate with other KAM residing on other domain.

- Ontology information of one KAM constitutes part of ontology of the domain it resides in.

3.3 KAM Model

The KAM architecture is based on a "society of intelligent agents" which collaboratively discover, markup, and organize relevant knowledge objects into a semantic knowledge network on a continuing basis. We can show it as Figure 3.1.

3.3.1 Knowledge Unit

The basic knowledge unit in KAM is a revision of Learning Object Metadata(LOM).[**?**]. It is a metadata defining all aspect of learning object. But most important features of it are "category, data type, relation, annotation, and category". We can draw a Jan object into following diagram,see Fig. 3.2.

In this diagram, we can see, two JANs within that the rectangle area belongs to a same class. All threes JANs in this diagram belongs to a same category. Each JAN has its annotations and relationships with other JANs. This representation requires that JAN should be unique. If we use URI to identify a same knowledge object, the URI of JAN in different KAM should be different. This satisfies the premise of URI. To organize URIs, a Resource oriented Architecture suits all requirements well. In this architecture, all resources are identified by URI and be organized according to a logic view. In addition it simplifies the API of KAM web interface into basic HTTP operation. This is illustrated in Chapter four.

JAN as the basic knowledge unit in KAM is extracted from various type of sources. In KAM, we focus on three important areas in our daily life: personal file system, personal bookmarks and personal email system. All JAN used in KAM area extracted from these three sources. The detailed exaction process is illustrated in Chapter five.

3.3.2 User Based ontology

Ontology plays an important role in KAM. As we can see from scenarios of Dr. Watson. When he found out some valuable resource, he would like to add into his personal knowledge network and also share with other colleagues. His knowledge network which right now does not exist in a visible format, purely in his mind. If Dr.Watson can convert his mental knowledge network into a concrete format, then his ontology is generated. if a KAM knows Dr. Watson so well, it can help Dr. Watson to create his personal ontology by creating taxonomy from his interesting areas and JANs from all resources Dr. Watson saved as his personal data. So to some extend, KAM can help Dr. Watson to reorganize all his knowledge into a visual format.

3.4 KAM architecture

KAM works with AI agents, and facilitates users' knowledge acquisition. It is the detailed implementation of KAM concept.
We define the KAM model as Table.3.1:

3.4.1 Taxonomy and the Semantic Net of Knowledge (T and R)

The first step in organizing any knowledge base is classification of the constituent Jans and interlinking them to form a semantic net. This requires that we first define a taxonomy that is appropriate for the domain. From a user perspective, this is aimed to build up user ontology into hierarchy architecture. The user ontology is different with the user profile. The prior concerns the classifications and their relationship. The later focus the user behavior model. Also because of the similarity of users, it is necessary to create Group Knowledge network that allow user to share JANs among groups.

Table 3.1: KAM Model

KAM – X	=	
pKN	=	Personal Knowledge Network
gKN	=	Group Knowledge Network
pON	=	Personal Onytology
CN	=	Context Network
iAPP	=	Intelligent Apps
pFS	=	Personal File System
pBM	=	Personal Bookmarks
pMS	=	Personal Mail System
dA	=	Discovery Agent
mA	=	Markup Agent
iA	=	Linking Agent (or Organizing Agent)
cA	=	Context Agent
vA	=	Visualization Agent
pA	=	Perusal Agent
sA	=	Search Agent
iA	=	Integrity Agent
oA	=	Ontology Agent
X	=	the domain name

3.4.2 The Discovery Agent (d_A) and Markup Agent

We have adopted the agent paradigm to describe all the services needed in building or using KAM. This may be viewed as a human-program continuum where a service may be provided manually or automatically by invoking a program using the concept of mixed initiative - where a program or a human depending on the circumstances may initiate an action. In our initial version of KAM, the discovery agent is a human who goes through normal search processes discovers a useful JAN which will be "marked-up" (to assist in classification) and submitted to a KAM builder where it will be classified and organized into the targeted knowledge network. The current version automated this process. This discovery agent actively search knowledge unit in the user space along with the global space.

The query result in turn was analysed by Markeup agent. The one with highest similarity would be submitted to the organizing agent.

3.4.3 The Organizing Agent (oA)

Once a Jan is obtained by “clicking” on the “markup” button (installed by a user on his or her browser) or through the KAM client interface, it is handed-off to the organizing agent which first ensures that the JAN represents a genuine link (one that is not broken or submitted by an unreliable source). It then examines the markup information, which is used for classification and interlinking. This information is also used to creating inverse links.

3.4.4 The Consistency/Completeness Agent (cA)

The Consistency/Completeness agent is mainly responsible for ensuring the integrity of the KAM by removing broken links and Jans that have been designated as inappropriate or irrelevant. The cA also will change the color of nodes whose semantic links have not been set for lack of information. This color-coding can assist the contributors of Jans to look for the needed information and set link values thus making the knowledge network progressively more complete.

3.4.5 The Visualization Agent (vA)

The visualization agent is responsible for displaying the KAM knowledge network in a variety of forms based on user preferences. One of the most useful ways to display a knowledge network is by drawing a hyper tree or as a radial graph where nodes at each level at the end of radii are equally spaced. The user can browse the lower level nodes by “clicking” on the nodes as the information at lower levels unfolds. In this view, the user gets radial graph view of the whole knowledge network from which the user can navigate to the area of interest as it unfolds with more and more detail as you near the target. In addition to the radial graph model, we intend to provide a variety of other representations including hierarchies and network traversal following predefined patterns.

3.4.6 Ontology Agent

The Ontology agent is responsible for importing and exporting User personal ontology created by KAM. It supports the OWL standard. The exported file can be read by tools supporting this standard, like protege. Also Ontology can help user to grow their knowledge network by importing other people's ontology file.

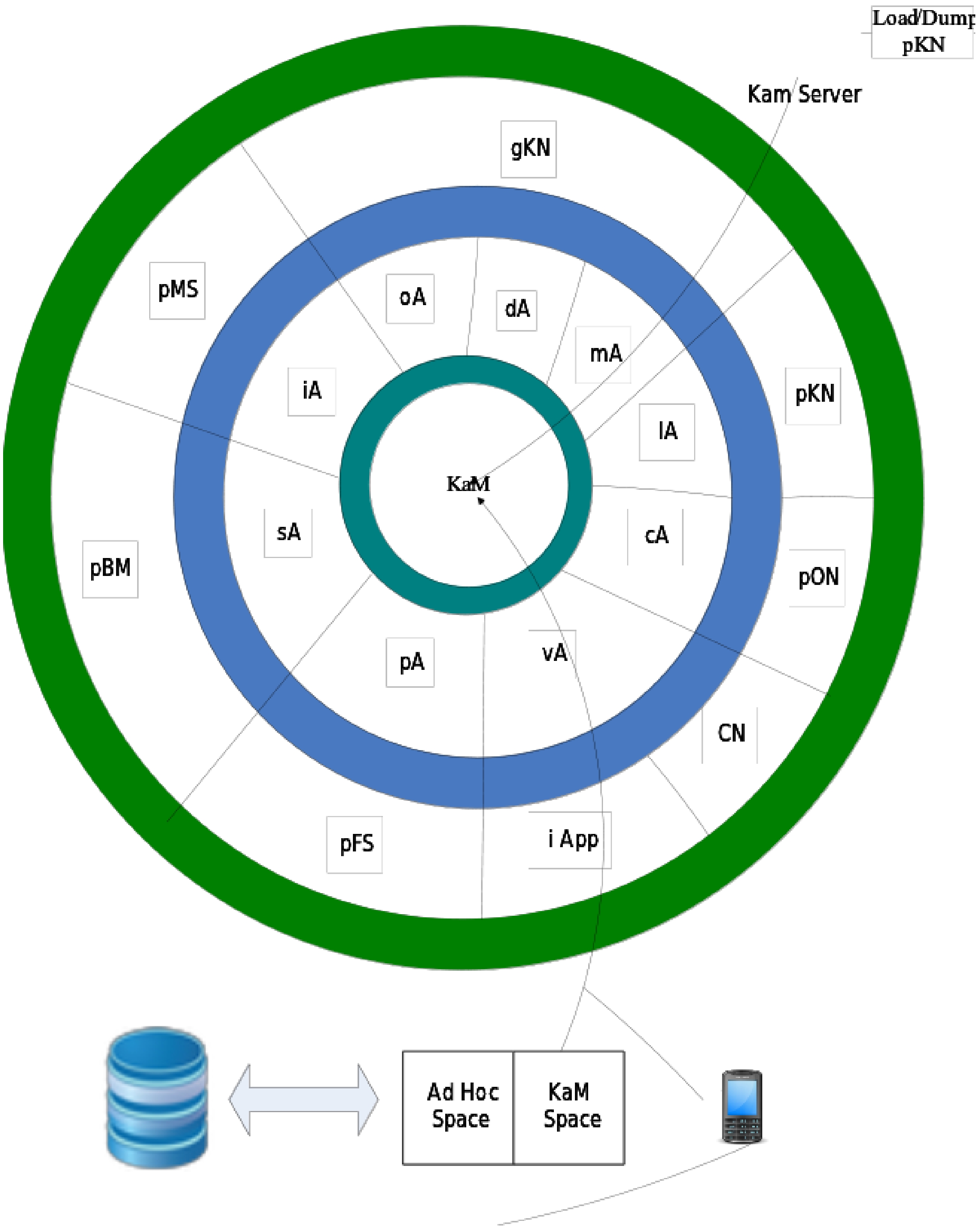


Figure 3.1: **KAM at a glance**

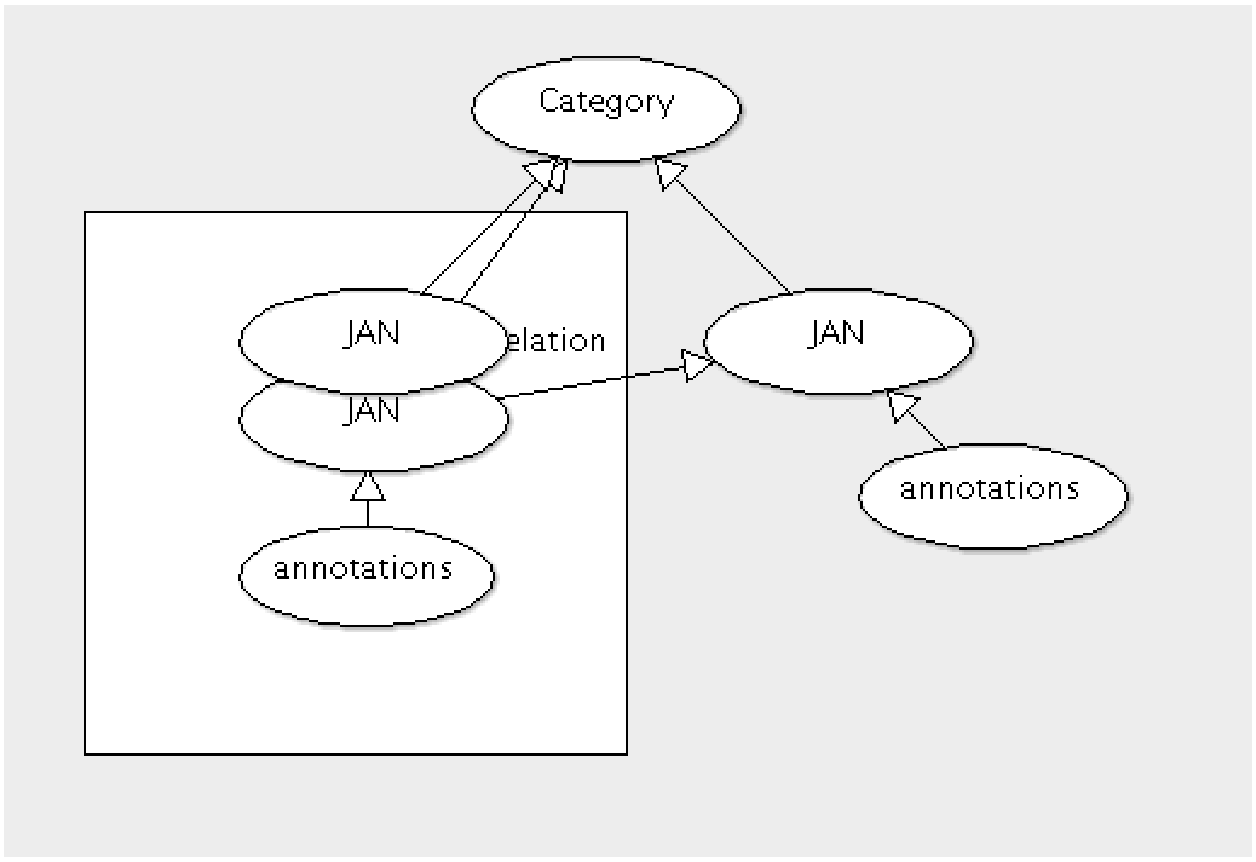


Figure 3.2: **A JAN Object**

Chapter 4

RESTful architecture

In this chapter, I illustrate the implementation of ROA in KAM. The ROA structure resolves the requirement on the uniqueness of resources in KAM. And the stateless feature of ROA eliminates the difficulties in organizing the knowledge concerning on the user context. All operations are simplified into basic HTTP actions.

4.1 Introduction

RESTful web applications rely on named resources in the form of Uniform resource locators (URL) and Uniform Resource Identifiers (URI) which are different from the information encapsulated in Simple Object Access Protocol(SOAP) message as individual XML files. Everything on World Wide Web is a resource and the reliance of the Rest API on named resources instead of messages eases the retrieval of a specified representation of the resource. Meanwhile the Rest API treats the resources as “nouns” that can be uniquely identified by a URI. A request associated with resources (noun) should be treated as a “verb”. An example of this usage is to GET a document identified with a unique URI. The use of URI’s enabled the discovery services without them being published to the centralized repository apriori. The naming strategy ensures that the REST API acts as the simplest HTTP requests and responses to send and receive information to/from applications. The following operations are supported:

- Retrieving Information (GET)
- Modifying Information (PUT)
- Creating Information (POST)
- Deleting Information(DELETE)

To explain the RESTful Operations with an example, consider a URI: http://Vijjana.org/users. Table 4.1 shows the basic operations and their corresponding meanings..

Table 4.1: RESTful Operation

Verb/Operation	Application Task	Explanation
GET	Read/ Retrieve	If a get request is sent for the URI it retrieves all resources related from this URI
PUT	Create/Update	When there is no resource related with this URI, then create new one, otherwise replace the old one with new one.
POST	Create	Create one resource related with this URI, the return should be a representation of this resource
DELETE	Delete	Delete the resource related with the URI

4.2 RESTful Features

One important feature of RESTful web services is their statelessness. A stateless web application requires the client to submit complete, independent requests. When processing this complete and independent request the server doesn't need to retrieve any application context or states.The design and development of the entire system is therefore arguably simplified. The lack of states of the intermediate servers completely avoids the synchronization issue between the session data and external applications. The state information is, on the other hand stored in the client. The overall response time of the system also benefits from the simplified server applications. One drawback of the stateless design is the server needs to receive repetitive data sent in a series of requests, to simulate the otherwise not maintained connection information for established clients. The requests with the repetitive information is sometimes referred as “overloaded post”.

4.3 Why KAM uses RESTful

KAM is aimed to establish a framework that facilities the construction of personal knowledge networks. KAM claims that a user crosses several domains of interests. To build the user profile, we rely upon Ontology, which defines a series of classifications and also relationships among the classifications. ROA architecture is a best candidate that fits the hierarchy requirement because the resource in ROA is purely and consistently identified with URIs.

4.3.1 Hierarchy Structure

In contrast with other resources, knowledge is abstract. To better interpret them, knowledge is often classified into categories, a process in which knowledge units are enriched with hierarchy information. Several resource websites, open directory [?], for example, already represents an enormous amount of knowledge units with well classified hierarchy information. The meta information fulfills the assumption of Resource-Oriented Architecture[?] (ROA) and also simplifies the resource retrieval problem. Meanwhile, according to the Web 3.0 standard[?], knowledge units, as a type of information, is to be retrieved upon semantic relations, which, to some extent, are extracted from hierarchy information.

4.3.2 Consistency of URI and URL

The knowledge units collected by a KAM system are mostly Internet URLs browsed by the users. One URL may be identified with several users, but in different, user specific perspectives. Meanwhile, people may contribute published books or articles as JANs into KAM, too. ROA is the best choice for KAM. The RESTful service would promote the efficiency in information retrieval . For RESTful services, resources are regarded as independent server resource objects, providing services for retrieving information. To explain this, we can look at one example. For instance, to find out the knowledge units about term "sportswebsites", previously we have to run a whole database scan to retrieve related information. If it related to a particular user, the query is executed again but confined with user information that might relate to certain context. However, in ROA, the naming strategy reassures us from this pressure. We still maintain one user-favorite table. However, the URL is examined when we make the HTTP POST operation. For instance we can send a get request to the URL http://Vijjana.org/sportswebsites. The server resources defined with this URL would take action. The HTTP response sent back should contain a list of resource units. All these resource units are collected from users knowledge network, which are related to the term "sportswebsite“". All these knowledge units are also Uniformed Resource Identifier, such as "jim:url:http://espn.com". So when we need to get a particular user knowledge about "sportwebsite", the results can be easily parsed by visiting http://Vijjana.org/sportswebsites/jim where will return the sports websites user jim saves. This also would benefit us on status check that is to make sure the information is correct. To do this, we ran a consistency agent to constantly remove entries where their corresponding JANs are no longer valid. Because the all the resources are URI, we avoid a lot of duplication checks when considering same URL are saved by multiple people.

4.3.3 Resource Security

As we explained before, one feature of RESTful services is their statelessness, accepting clients request without saving the state information. For a user-based system, however, if the server doesn't maintain a user state context, problems may arise. On the other hand, heavily relying on a persistence connection to maintain the state consistency is prohibitively expensive. The problem is solved in RESTful with security schema. This is helpful in KAM for user to create personal cyber world with privacy concern. The private space and public space are defined and controlled by the write/access privileges on the resources. From an outer view, the whole space is readable and transparent. However from inner view, only authenticated user can POST or PUT entity. All the user related operations are distinguished by the URI structure.

4.3.4 Other benefits

Another important feature implying in RESTful is robot-driven. This fulfills the feature proposed in Web 3.0 standard. A knowledge system should be driven by machine instead of human contribution. The strong service layer plays the important role in enabling this feature. It also provides a way for resource exposing. To some extent, KAM is intending to open service for others. Also it provides a way for KAM to become to be a collaborative platform.

4.4 Resources in KAM

The basic knowledge unit JAN is the core resource in KAM. Also user, as the contributor of JAN, becomes to be another important resource in KAM. A normal person acts in several roles in his/her daily life, like roles in family, roles in job, roles in friends. All these roles vary with the age grows, environment changes and all other possible factors. Considering on this, well-structured context information would benefit us in defining user URI.
Also related terminologies are also resources here, such as following:

- subscription of update of one user knowledge network;
- consistency check;
- reorganize knowledge network;

All these terminologies are mapped with certain actions. When one terminology resource is called, its corresponding action would be taken and may trigger the state transition for other resources.

4.5 RESTful Service for Work Flow

Imagine the following scenario, when a user, Jim, accesses the KAM system with RESTful API, Jim first accesses the entry point which is an index page listing all sorts of services provided in KAM system. Afterward, when Jim may choose to log in his space, where saves his personal knowledge network. Whiling surfing the Internet, Jim may find out interesting sports news and would like to create one JAN for it. The Vijjana-Addon [?] will help Jim to markup the URL and generate the meta data for this JAN. After this, Jim makes a POST request to his knowledge network URL by adding this JAN into the knowledge network. Then this JAN has its own URI and also available for AI agents. The whole process is illustrated in the following diagram , Fig 4.1.

APIs are required for the following scenarios.

- entry point for both the whole system and user;
- user knowledge network
- user knowledge network node: JANs

Based on the above scenarios, we design the APIs and list them in Table **??**, as following:

4.6 Detailed Implementation

RESTful service does not change the framework of KAM illustrated in[?]. It appears in the KAM framework as part of middle layer and help to facilitate the Web user interface. We can abstract the KAM framework into several parts as Fig. 4.2 represents.

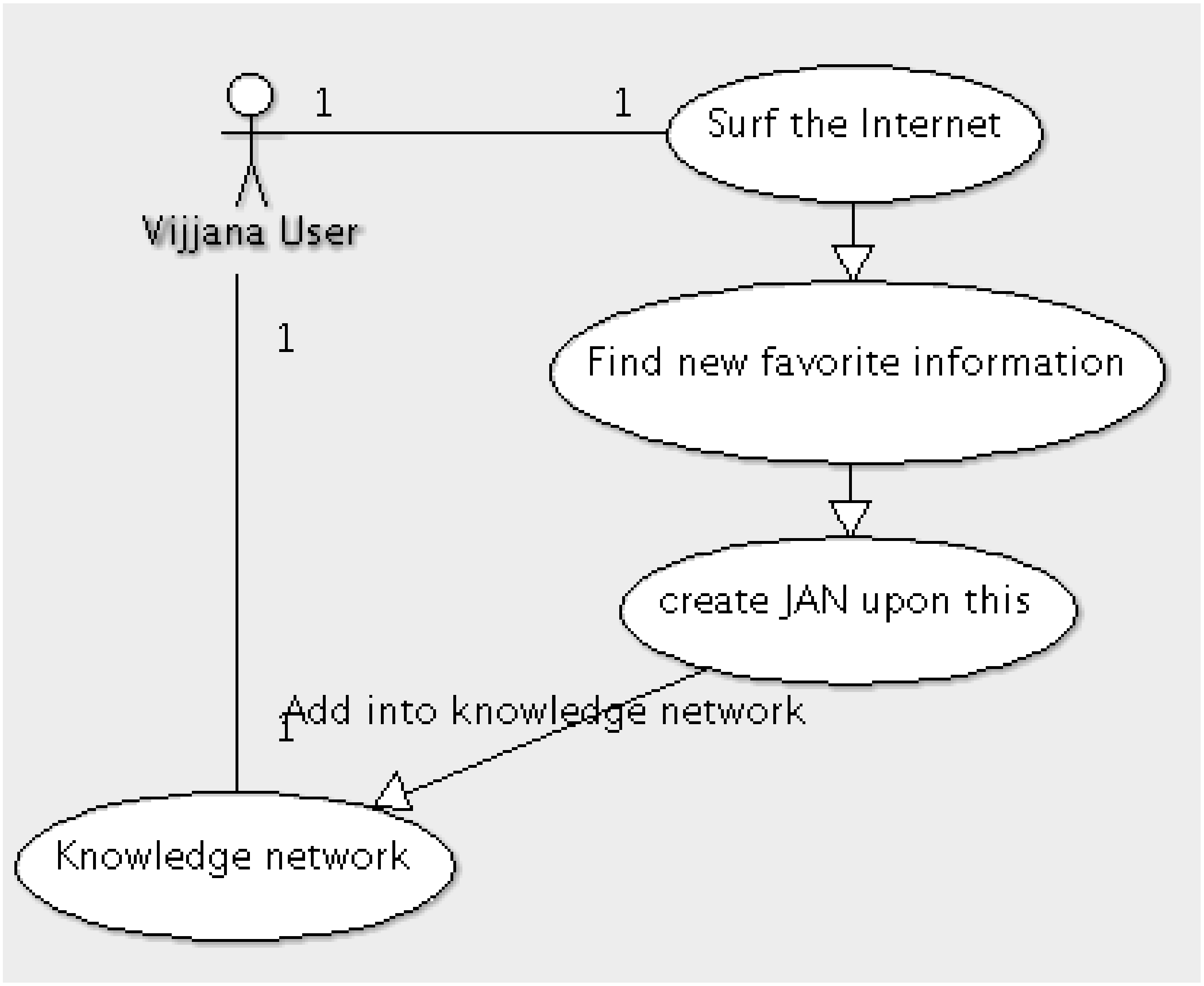


Figure 4.1: **A common work-flow for a typical user**

4.6.1 KAM framework in ROA

From Fig.4.2, we can tell that the RESTful service acts as an important role in KAM system, presenting in the intermediate layer to work with agents and Core KAM module. The Agents are responsible for all intelligent tasks, like creating JAN upon specific URL. It cooperates with the all modules and would reflect result into data store. The Core KAM module takes the role to store/retrieve JANs into/from data store. To cooperate with others, the RESTful service then reacts upon the requests from the Core KAM module to represent JANs into Web layer, and also communicates with agents to create or organize JANs.

Table 4.2: API Table

URI	Explanation
/	Entry points for whole system
/user/ALL	Entry point for all users
/user/jim	Entry point for jim
/user/jim/jans (GET)	Retrieve all JANs contributed by all users
/user/jim/jans/add	a new jan form
/user/jim/jans/{janid}/{POST}	POST an new JAN for jim
/user/jim/jans/{janid}/edit	update one particular JAN for jim
/user/jim/jans/{janid}	Get a JAN identified by id from jims knowledge network

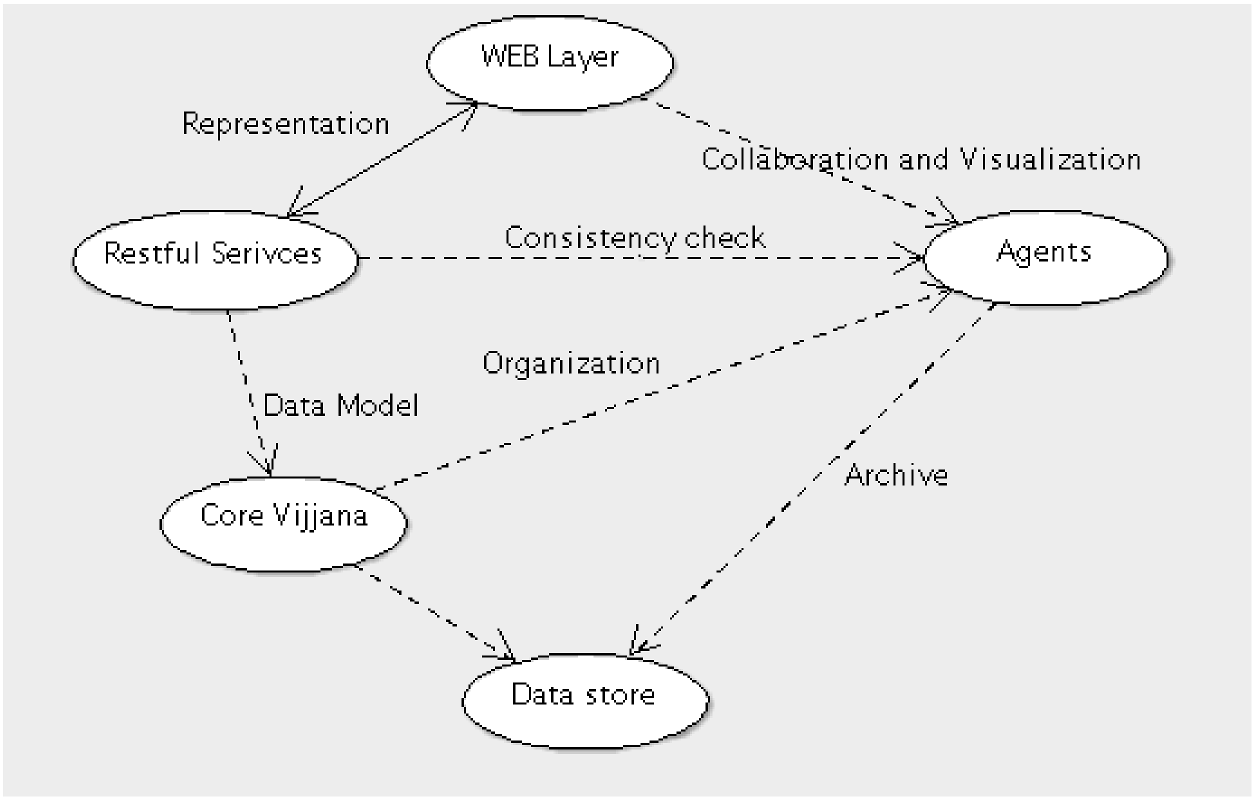


Figure 4.2: **Vijjana framework**

4.6.2 Data format

As mentioned above, in KAM, we need different type of format to represent our knowledge unit. For example, the graphML , a special graph xml format, is generated as our visualization input. The representation format, such as json, xml and graphML is also represented according to the request entry of our restful layer. When the object is POSTed, the corresponding serverResource would transform the object into JAN without any format. When a JAN was requested with format requirement with the url according to this pattern {JANID}{format}, the corresponding format data representation will process and generated. For example,/jans/892/graphml would generate the GraphML format of JAN with id 892. Internally we rely on Spring marshalling method to create an xml format and JSON serializer to create JSON format.

4.6.3 URI resources and compared to Session

In the RESTful framework, all things are concerns with URI without concerning on the session information. For example, in our implementation, all the operations related to user are related with URL http://vijjanaweb.vijjana.org/users. The underneath operations are the basic operations of HTTP. When register a new user, it submits a POST operation to the URL "" and then one server resource responses with store this user. When we visit the URL "http://vijjanaweb.vijjana.org/users/john". Another server resource will provide the service instead of the previous one. So even the two URL share the base URL but its underlying structure is different. Compared to the session related server, if we need to retrieve a user information, then the client needs to initiate a request to server with query string "username=john". To response this, the server will open one session for this request and the corresponding servlet binding with URL like "/users?username=draft" will response. Then the response returns to client. Here the service resources binding with "/users" will response on any use request, acting as an intermediate station to dispatch response correspondingly. Obviously, the RESTful implementation illuminates this with different URI resource binding, which form a one to one mapping.

Chapter 5

Vijjana Keyphrase Extraction Algroithm

In this chapter, I will illustrate the key phrase extracting algorithm used in the vijjana. From chapter two, we can see that the the semantic word concept plays an important role in knowledge filtering and user profile construction. Here I explain my method for extracting keywords. I will explain its usage on the user ontology and knowledge unit in the next chapter.

5.1 Background

Keyphrases extraction, as an information retrieval technology, plays an important role in information indexing, clustering and inferencing. Accurate keyphrases depict content at a high level abstraction and are also used to define a semantic index in a knowledge based network. With the development of search engines and the semantic web, tons of information accumulates at a rapid speed. Most of it is disordered and is represented without an obvious relationship. To reveal the inherent relationships and make corresponding classifications, researchers in machine learning impose statistical models that rely on techniques like the naive bayes[?] theorem, decision trees [?], and example-based models [?]. Most of them can achieve very accurate results with domain-specific knowledge which ensures that the conditional independence assumption in these statistical models is true. However this assumption is rarely true in real world applications, especially in a real time application which provides concurrent feedback with user input. To conquer this, Artificial Intelligence techniques are introduced into the process. They can change their actions with the awareness of context status changes. On the other hand, what they sacrifice is having additional run-time cost in adaptively re-run the model to produce higher quality results. To seek a balance between statistical and AI methods, we can combine their features. From the AI perspective, adaptive learning is performed the confines of a small window, usually a region of a certain word-length. Heuristic selection, as the basic AI operation, is conducted based upon the accumulative learning on these certain-length regions. For a particular paragraph, the final result of this method is a list of keywords which can stand as a summary for the whole paragraph's content. In order to better organize the accumulative learning, we need to apply statistic analysis in investigating the functional factors that largely affect the results. In general, several factors could potentially impact result accuracy and algorithm performance. These factors range from category factors like specific domains to quantitative factors as text lengths. The earlier researches[?][?][?] in this area focuses on changing the amount of training data or the number of keyphrases to diagnose the impact of such factors. Their research results suggest that the training data set size has little impact on final keyphrase accuracy after its quantity exceeds 20. Thus, the quality of the training set influences results greatly as compared to the size of the training set.

In the KAM model, information inference is a key factor in discovering knowledge units, revealing connections between users taxonomies and linking ontologies. The user input document is interpreted into keyphrases that would be used in information indexing and clustering. These keyphrases are also analysed in a context model to determine the current user context. All of these determine that our algorithm needs to be accurate and relies less upon the algorithm's training process. To solve the problem, we utilize a Markov chain Monte Carlo method to sample selected paragraph snippets. Based on sample distribution and maximum entropy theorem, we designed our own keyphrase algorithm. The assumption of this algorithm is that the keyphrase, regarded as a characteristic word, discretely appears continuously in each equal size region of a document. In this rest of this chapter, section two presents related works in this field. Section three gives the theoretical background of our algorithm. Section four focuses on illustrating the algorithm details. Section five examines several factors in which we have interest.

5.2 RELATED WORK

In the English language, people use different phrases to distinguish a given time with with the current moment. This brings in uncertainty and word-polymorphism problems in text mining. To solve this problem, several stemming algorithm are proposed. Lovins[?] and Porter[?] are two popular algorithms for stemming English words whose alphabet consists only of 26 characters. Stemming is important in keyphrase extraction algorithms which call for accurately recognizing minor differences between words. Krulwich and Burkley[?] proposed to use heuristic rules to extract key words in 1996. Their method largely depends on syntactic rules. Adam L. Berger and Vincent in 1996 [?] proposed to use the maximum entropy for natural language processing. Munoz proposed a method of generating keywords by using Adaptive Resonance theory (ART) neural networks [?]. The keywords generated by this algorithm have a low precision and the algorithm also doesn't fit for more-than-two-words keyphrase. Frank et al. (1999) developed a keyphrase generating tool called KEA which uses the Bayesian approach as a learning procedure[?]. This algorithm largely improved the accuracy by heavily relying on well structured training data sets gathered from a specific domain. Andrew and Dayne at 2000 proposed the MEMM model[?] to tag sequential data, which utilized a theory similar to ours but relied on learning data and specific constraints.

All these researches reveal that the heuristic method is much more effective than analyzing patterns lying between sentences. A Large amount of training data would greatly improve results but also sacrifice a shorter running time in gathering them. However, almost all of these algorithms put much energy into discovering and exposing very small factors which impact performance. Frank and his/her colleagues in [?] suggest that well structured domain characteristic data would improve results and also reveal their algorithm's result would have no improvement after trained with 50 documents on a specific domain.

An overall goal of our work is to propose the algorithm and also to identify significant factors from several general ones. Our algorithm is independent from training data.

5.3 Theoretical Background

A standard document consists of thousands of words. After it is stemmed, each word will be identical and also have its own distribution. If we define a *information set* X as a set of words, then the distribution of this information set becomes a multinominal distribution with respect to the distribution of a random variable x_i with probability p_i, where x_i stands for a particular word in this information set.

$$f(X) = \begin{cases} \frac{n!}{x_1! \ldots x_k!} p_1 \ldots p_k & \text{when } \sum_{i=1}^{k} x_i = N \\ 0 & \text{Otherwise} \end{cases} \tag{5.1}$$

where $x_i \in X$.

As mentioned before, the keyphrase, regarded as words distilling the main idea of a whole document, should appear in the document in a discretely in a uniform distribution.

5.3.1 Term Weighting

The importance of a term in a document is revealed by its contribution to the whole document. To evaluate the contribution, it is necessary to formalize them in a numeric fashion. The traditional method is to construct the document into a vector $d = \{w_0, w_1, \ldots, w_n\}$ where the w_i denotes a unique word that has appeared in this document. For each word, w_i, the weight is calculated by its frequency in this document. The intuitive and easiest way is using a term occurrence frequency in the document as term frequency. There are two other types term frequency [?] augmented normalized term frequency and binary term frequency. The binary term frequency is also as simple as the occurrence frequency. It is useful when there is an existing training set. If a term appeared in the training set but not in the document, the term frequency is set to zero. If it appeared in both, then the term frequency is set to one. The augmented normalized method is a variation of binary frequency but independent of a training set. It adds an augmented factor and normalizes the term occurrence by dividing the maximum term occurrence. It is defined as $0.5 + 0.5 * \frac{tf}{tf_{max}}$ where the tf is the occurrence frequency and tf_{max} is the maximum term frequency in this document. In our VKE algorithm, we use the occurrence algorithm as our term frequency. In our context model which will be illustrated in the next chapter, we use the augmented normalized term frequency. The reason we use the occurrence frequency is from the consideration of the term density in our random sampling process.

5.3.2 Metropolis-Hastings algorithm in sampling

The random selection in our VKE algorithm is ensured by the Metropolis-hasting algorithm[**?**]. It walks through the whole document and generates "candidates" from the proposal distribution which is a multinomial distribution of the whole document. This algorithm is simple but quite effective in generating samples from a continuous space. It involves an Acceptance-Rejection progress. For a given item x, it has a density $\pi(x) = f(x)/K$, where f(x) is the unnormliazed density and K is a normalized factor which might be unknown. If we already know there is a density, h(x), that is generated from a known method and suppose there is a known factor c such that $f(x) \leq c * h(x)$, then we can obtain a random variate from π.

This original algorithm is described as following:

```
1: initially, random selected one item $y$ as start point.
2: **for** $i$ =1 TO sample size **do**
3:     proposal step: generate "candidate" $x$ from the given distribution Q, and $mu$ from
       U(0,1)
4:     $x \sim q(x|y_{i-1})$
5:     **if** $\mu < \alpha(x, y)$ **then**
6:         $y_i = X$
7:     **else**
8:         $y_i = y_{i-1}$
9:     **end if**
```

```
10: end for
11: return
```

The A-R process is determined as $\alpha(x, y)$. If $u < \alpha(x, y)$, then a new candidate is obtained, otherwise it stays. Therefore, the transition from x to y according to $alpha(x, y)$ can be described as following:

$$P(x, y) = q(x, y)\alpha(x, y), wherex \neq y \tag{5.2}$$

One of our M-H round shows result as :

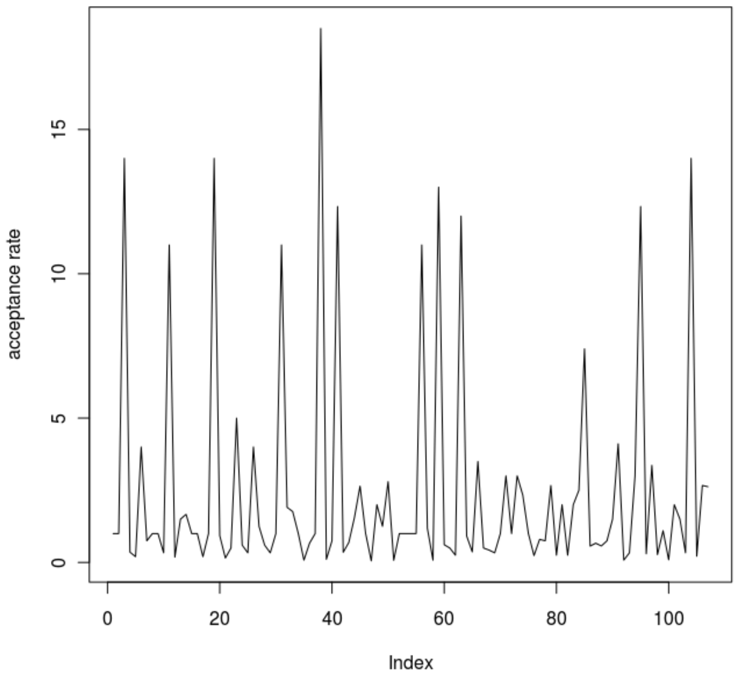


Figure 5.1: Metropolis-Hastings

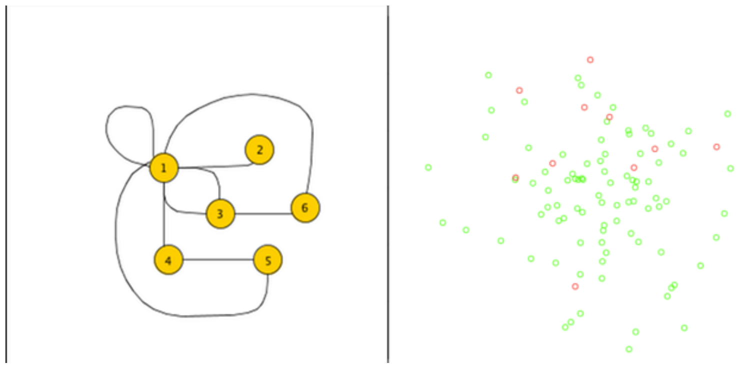


Figure 5.2: Monte Carlo Approach

5.3.3 Monte Carlo Method in VKE

The traditional monte carlo method is used to sample i.i.d data set X with a known density $p(x)$ on a space where the posterior distribution is defined. This is shown as Fig. 5.2 For real time applications like text processing, the traditional method is not applicable. Correspondingly, the sequential monte carlo method [**?**][**?**] is devised to solve this problem. It is required to satisfy the following criteria: 1). It has an initial distribution, which is non-zero but can be arbitrary generated and 2). The latest information should satisfy the support of posterior p(X) or even greater than the earlier information gained in the process with respect to state Y.

$$p(X_0) > 0 \tag{5.3}$$

$$p(X_t|Y_t) > p(X_{t-1}|Y_{t-1}) \tag{5.4}$$

To better interpret these requirements, we can consider the whole process as set transitions. A generic transition is related to one past state Y_{t-1} with probability $p(X_{t-1}|Y_{t-1})$, current state Y_t with probability $p(X_t)$, and one observation o_t gained from current state. Correspondingly, at the very beginning, the initial set is stateless with a non-zero density mass $p(X_0)$. An information set is collected at this state. Meanwhile the first observation o_1 is computed from this information set based on $p(X_0)$. This observation leads to a state transition, in which new information set is collected and new observation is computed. This process is displayed in Fig. 5.3.

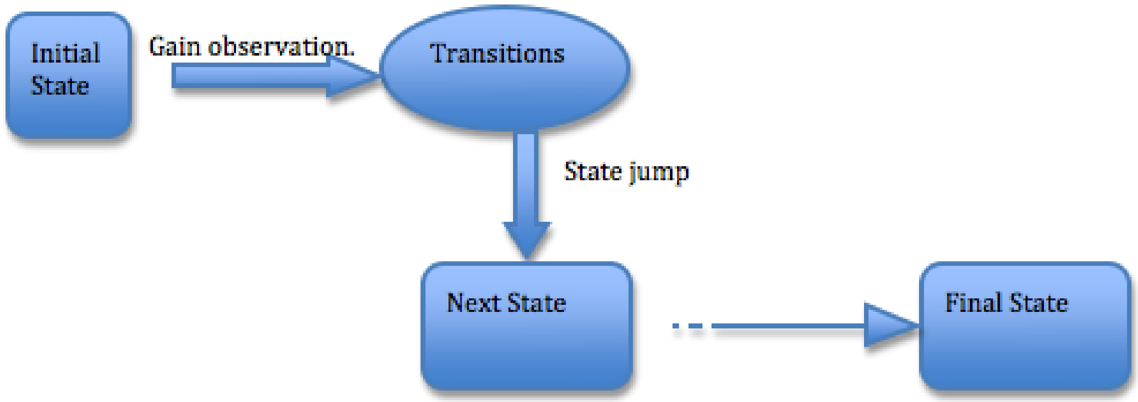


Figure 5.3: State transitions

An important property to mention in a Markov chain is the aperiodicity. Any state transition needs to avoid becoming trapped in a loop. This property is also important even in real time applications like sequence text processing where the data is cached and computed. When the state transition only moves from a state with lower probability to a state with a higher one, the target distribution would be shaped in a gradient way. This mechanism is a revision of MCMC algorithm, called Hybrid Monte Carlo.

5.3.4 Maximum entroy theorem in VKE

Entropy is a concept introduced into information theory from physics. It is used to be a measure of disorder. One feature of entropy is that it explains the spontaneous process. Here we can simply explain entropy by a simple example when it applies in text processing. Suppose one sentence consists five words: A, B,C,D and E, in which they may appear more than once. From this sentence, two information sets are generated. One contains three words, A ,B and C,and the other is formed by A,B,C,D. The intuitive density is as following:

$$p(A) = p(B) = p(C) = p(D) = p(E) = 1/5. \tag{5.5}$$

This naive model treats every possible candidate as having equal probability, however they do not since they appear in the sentence more than once. From the two information sets, we

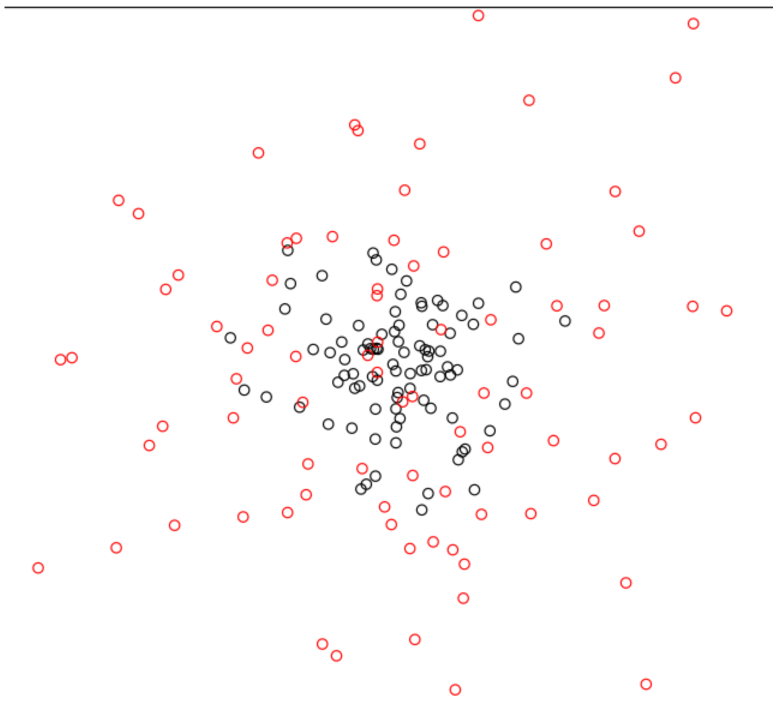


Figure 5.4: Entropy Concept

may think the first information containing words A, B and C may be more appropriate in depicting this sentence. However, to verify this, we need to prove these words have a more uniform model than the other two. To prove this, we can use the entropy concept.

The definition of entropy in information theory is given in term of probability of message delivery. Messages can be lost during the transmit process. For this reason, the probability is regarded as a success rate. For an information set, the entropy concept is used to be a measure of how much information was in this information set. See Fig. 5.4

The definition given can be interpreted in the following way to fit in our keyphrase extraction circumstance: the entropy of a word in a document is a measurement of to what degree this word can stand for the main idea of this document. Then the information set which has the largest entropy value should be regarded as the best keyphrase set. The new definition fulfills the features of the entropy concept and also suits well for our complex problems that we are trying to solve. As same as the formula defined in information theory, we define the entropy formula as :

$$E(IS) = \sum_{i=0}^{N} P_i(X) \ln \frac{1}{P_i(X)}, \forall X \in IS. \tag{5.6}$$

Where the X denotes a particular word, $P_i(X)$ is the possibility that the word X shows in the ith certain transition. It is the ratio betwen its number of it appearances in a list of frequently-appearing words and the list size in a particular Monte Carlo Set . N denotes the total number of transitions which also relate to region numbers that the document is divided into. Our algorithm selects only one transition per region. So totally we have N Monte Carlo Sets.

From the definition we can say the words with the largest entropy values are more likely to be keyphrases. This is also our final conclusion drawn from the entropy concept. However the premise of using the entropy concept here is the the equilibrium of keyphrase density in document.

5.4 Algorithm design

The VKE algorithm processes a document in three steps. The first step is the preprocessing step. It is to preprocess the given document. This step includes two minor steps. 1) The User predefines three important variables. One is a variable defining the threshold of term weight. Another is a region number that is used to determine into how many pieces a document should be divided. The third variable is the maximum trail number that indicates the maximum number of state transitions permitted in the VKE Model. 2) Generates two tables: a term weight table and a term location table.

The second step is the HMC heuristic selection step. In this step, based on the regions we state, we first define the HMC state that is a HMC selection would perform on this particular region. For each state, we label it as either a high term weight state or a low term weight state according to the terms frequency we calculated on step 1. Then We apply the M-H algorithm on all states. The M-H algorithm will generate a Monte Carlo set on each state that stands for the current observation. Based on the observation, we leapfrog only if the current monte carlo set satisfies two conditions: the mean value of term weight calculated from this set is (1) larger than the one from previous state and also (2) the ratio of the two mean values is larger than a random value selected from a uniform distribution, U(0,1). The transition can only move to a higher term weight state. Till all high term weight states are all visited, HMC algorithm finished.

The third step is the maximum entropy calculation of the selected HMC observation.

5.4.1 Preprocessing

In the preprocessing step, a user-defined threshold for minimum term weight is required. It is usually a percentage factor, valued from zero to one. This factor is multiplied by the maximum term weight to get the real threshold. The other user-defined value is region number. For a given document, we need to construct two information tables. One is the token-length table $L(pos, w_{pos})$. The other is the term weight table, $W(term, tf)$. $L(pos, w_{pos})$ which describes the document length continuous space. Both the M-H and HMC heuristics need location information from it to perform the random walk. Table $W(term, tf)$ laid the foundation for the VKE algorithm. All the calculations of term density and entropy are based on the term weight table.

5.4.2 M-H in VKE

For a M-H algorithm implementation, it is important to specify a target density where the random variate generated from. In VKE, we specified the term density as our target density. For a given document, the term density is described by term frequency in a document length space as $\pi(w_i) = \frac{tf_{w_i}}{L_d}$ where the tf_{w_i} stands for the term occurrence and L_d is the length of the given document. Then A-R acceptance method,$alpha(w_i, w_j)$, is defined as

$$\alpha(w_i, w_j) = min\left(\frac{\pi(y)}{\pi(x)}, 1\right) \tag{5.7}$$

Our M-H implementation is as same as the above section with specifying the density function.

5.4.3 HMC heuristic selection

The HMC heuristic selection is the essential part in the KEA algorithm. The region number determines how many states are in this HMC model. According to the region number, we divide a document into regions. The region information is calculated from the table $L(pos, w_{pos})$ that is generated in the preprocessing phase. Each region acts as an individual state. Based on the term weight information provided by term weight table $W(term, tf)$, we can label each state as a "high term weight" state,$S - htw$ or a "low term weight" state S_{ltw}.

A Monte Carlo set containing keyphrases generated by the M-H algorithm on a certain region is regarded as an observation from the corresponding state. The probability of the observation is calculated by high frequency term percentage, $P(X_i) = \frac{hfts_s}{T}$ where hf_s stands for the high frequency term size and T denotes the Monte Carlo set size. The State transition also involves an A-R progress. Different from the H-M algorithm, the acceptance condition is rigid and tough, which we need to satisfy the HMC premise that every leapfrog needs to be from a state with a lower probability to a state with a higher one. Therefore in our model, for a state transition, the transition must be from a monte carlo set with lower probability to a higher one. In addition, we introduce a random value z simulated by a uniform distribution on range from zero to one. For every leapfrog, the probability ratio of the current state over the previous has to be larger than z. These two conditions ensure keyphrases that appear in previous regions may also appear in the following regions. This conditional leapfrog ensures our assumption that using entropy is correct. The initial state is random selected. The ending situation is that all high term weight states are visited.

The process can be described as follows:

```
1: Start at region one with state $Y_0$
2: **for** i= 1 to N-1 **do**
3:     Create Monte Carlo Set $S_i$.
4:     Calculate $P_i$
5:     **if** $\frac{P(X_i|Y_i)}{P(X_{i-1}|Y_{i-1})} \geq 1$ AND $\frac{P(X_i|Y_i)}{P(X_{i-1}|Y_{i-1})} \geq \mu$ **then**
```

```
6:         Go to next Region
7:     else
8:         Redo procedure from line 3 to line 6
9:     end if
10: end for
```

5.4.4 Entroy computation

```
1: All candidates' frequency is computed
2: while Not Final State do
3:     Store candidates $p(x_i|Y_i)$ in an array $E_i$.
4: end while
5: compute maximum entropy from matrix $M$ consisted by $E_i$
6: sort M.
```

The correctness of this algorithm is easily proven. The concept of entropy means that the keyphrases are going to be spread through the whole document. From the view of keyphrase extraction, we are applying a syntactic rule on a well structured document. The heuristic method, Markov chain Monte Carlo (MCMC), provides the foundation for this assumption to be true by satisfying base possibility requirement.
The algorithm's run-time complexity is O(mn), where n is the length of document, m is a preset value of the maximum trials for fetching information set. The whole process will go through the document at most m times. If the document is large and m is comparatively small, then the complexity is close to O(n).

5.5 Experiment

5.5.1 Terminology

Before we proceed to explain our experiment, it is necessary to give several terms used in our study. Most of them are also general terms used in statistic modeling:

1. *Factors*: The variable that affects the response variable.

2. *Levels*: The possible value of a factor can assume.

3. *Response variable*: The measurement variable used to evaluate performance.

4. *Main effects*: The variance of response values caused by factor level changes.

5. *Interaction effects*: The relative changes due to two factors interacting with each other.

5.5.2 Experiment Design

Objective and Hypothesis

Our main objective as addressed before is to find out which factors have the greatest effect in impacting the results. Statistically speaking, we can try to find the factor whose main effect is obviously larger than the others with respect to its mean accuracy value. The second objective is that we are trying to identify whether or not there are some factors interacting with each other. Statistically speaking, we are trying to identify whether the interaction effect between two chosen factors are identical. Generally the Hypothesises that we are going to test are given as follows:

1. Main Objective:
   Null Hypothesis H_0: There is no differences amongst levels of a particular factor.
   Alternative Hypothesis H_a: There are differences within levels of a particular factor.

2. Second Objective:

   Null Hypothesis H_0: There is no interaction effect existing between two factors.

   Alternative Hypothesis: H_a: There are interaction effect existing between two factors.

Factors Selection

There are numerous factors which may impact the accuracy of keywords generated by the keyphrase extraction algorithm. Some of them are characteristics intrinsic to text documents. This demonstrates that some factors are beyond our control and lead our experiment to be an observational study. For these kind of factors, we classify the experiment units, which right now are documents, into corresponding factor levels by observing the distribution of their values. Some of them are algorithm parameters which can be varied by passing in different values. For these kind of factors, we change their values according to their factor levels. In our experiment, we select factors as Table 5.1 shows.

Table 5.1: Factor Table

Factor	Levels
document size	words number less than 30000, between 30000 and 60000, more than 60000
generated keyphrase number	5,10
domain specifics	domain1 to domain4

Hereby we provide justification for the factors and their levels used in this study.

Document size is the total number of words in document from which we extract keyphrases. As mentioned before, the size of a document as a document characteristic is also out of our control in our study. Thus we selected its levels by observing its distribution from our randomly-selected documents which stand for the whole population. In the end, we define its levels as three categories: less than 30000, between 30000 and 60000, more than 60000.

keyphrase number is the number of keyphrases generated by our algorithm. This factor is easily controlled since it is a parameter in our algorithm. Therefore we set its level as 5 and 10.

specific domain is a factor indicating which domain a particular document belongs to. It is also a characteristic of a document. We place the selected document into one of four domains from where we will later select it.

Response Variable

Since there is no measurement standard to follow in keyphrase extraction algorithms, we continue to use general methods that measure accuracy by the number of matches between machine-generated keyphrases and human-assigned keyphrases. In our study, since the number of keyphrases is one factor which might affect the result, we are expecting a larger number would generate more matches. The accuracy value we use is the ratio of the number of matches to the number of keyphrases as this equation shows:

$$Accuracy = \frac{Matches\ number}{keyphrase\ Numer} \tag{5.8}$$

One thing to mention here is that the final number of final matches is manually counted by humans. This may introduce validity threats due to human error.

Study Design

Random selection is a basic principle in experimental design. However, since a large amount of factors in keyphrase extraction algorithm study are also characteristics of documents, the random-selection principle usually also introduces some noise in sampled data. In our study, to largely eliminate bias, we randomly selected 24 documents as our sample data from four different domains. The diversity in this sample data lets us define their factor levels largely dependent on their distribution in sample data. This also fulfills the requirement of a balanced design. However it also means some factor levels are going to be variable which may bring in a random effect. We apply our algorithm upon these documents which are our experiment units. Since the interaction effect between any two factors is our interest, we have to use *Crossed Design* as our design model. Since the number of keyphrases is a factor that we have control over and it has only two levels, we can use this feature to refine our design as a *Randomized Complete Block Design* in which factor *keyphrase numer* is used as a block factor. This design lets factor *document size* and factor *domain specific* be crossed within the blocks of *keyphrase number*, as seen in Table 5.2. Since block design is used to largely eliminate noise, it would also be helpful in clearly identifying relationships between factors. Therefore we can consider the factor of the number of keyphrases as one block factor which would increase the power of Analysis of variance. In our study, the *document size* factor has three levels, and *domain specific* factor has four levels. So we have twelve treatments.

Data selection

To obey the random selection principle, we originally randomly selected 67 documents from five different domains as our sample data. However, when we applied the algorithm to them, we discovered that the sample data from domain one is largely different that the other four domains since the documents' sizes in it are too small compared with other four domains' data in which document size scale well in three factor levels. Due to this reason, we pruned the sample data set to be 24 documents only and from 4 domains to ensure the balance requirement. These 24 documents are fitted into 12 treatments because the keyphrase

Table 5.2: RCBD in VKE

k(5)	(D1,L2)(D1,L3)(D2,L1) (D1,L1)(D2,L3)(D3,L1) (D4,L2)(D3,L2)(D4,L1)(D2,L2) (D4,L3)(D3,L3)(D4,L4)
k(10)	(D1,L1)(D2,L2) (D1,L2)(D1,L3) (D2,L3)(D3,L3) (D4,L1)(D3,L1)(D3,L2)(D4,L2)(D2,L1)(D4,L3)(D4,L4)

is acting as a block factor. This means our sample size in each treatment is only two. This small sample size introduces bias. However, considering the cost issue, we continued to conduct our study with this small sample size.

5.6 Data Analysis

5.6.1 Analysis of Variance(ANOVA)

The statistic ANOVA technique is widely used in more than two levels of factorial design. The definition of it given in wikipedia is what follows [?], "In statistics, analysis of variance (ANOVA) is a collection of statistical models, and their associated procedures, in which the observed variance is partitioned into components due to different explanatory variables.". The detailed explanation of the ANOVA table will be clarified in the following analysis part.

5.6.2 General Analysis upon our objectives

Before we proceed to conduct the hypothesis test, we take consider the whole study first. From Fig 5.5 we can see there are no big differences among different domains; from Fig 5.6 we can see an interesting phenomena in that when the keyphrase number is five, the mean value is larger than the mean value when the keyphrase number is 10. We can estimate that the factor *klength* may have an effect of impacting the final result.

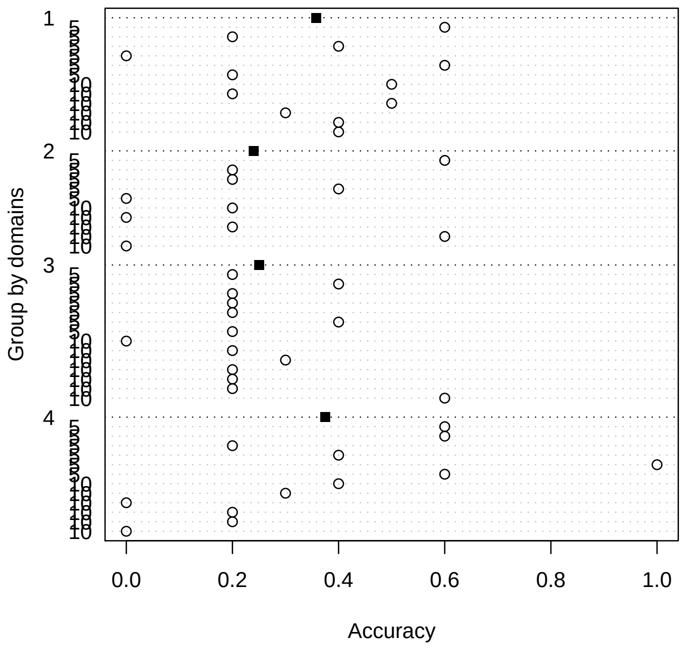


Figure 5.5: Overview dotchart 1: Accuracy Grouped by Domains

5.6.3 Hypothesis Test

First Objective: Main Effect

Recall the null hypothesis proposed in Section 3.3.1. We are interested in analyzing the main effect of factors that may impact the accuracy results and also their interaction effects. Our study is a three factor design. For each factor we, examine its main effect and also how it interacts with the remaining two factors. Then we construct our ANOVA test as following:

In this ANOVA table, a row indicates the effect caused by its corresponding factor which is listed at the first column in each row. *Df* means the degrees of freedom and this value is

```
> anova(lm(accuracy ~ klength + domain
 + tlevel + klength * domain
 + domain * tlevel
+ klength * tlevel))

Analysis of Variance Table

Response: accuracy
              Df  Sum Sq Mean Sq F value  Pr(>F)
klength        1 0.13021 0.13021  3.7227 0.06287 .
domain         3 0.17746 0.05915  1.6912 0.18922
tlevel         2 0.01051 0.00525  0.1502 0.86118
klength:domain 3 0.33484 0.11161  3.1911 0.03719 *
domain:tlevel  5 0.42641 0.08528  2.4383 0.05644 .
klength:tlevel 2 0.02442 0.01221  0.3491 0.70804
Residuals     31 1.08428 0.03498
---
```

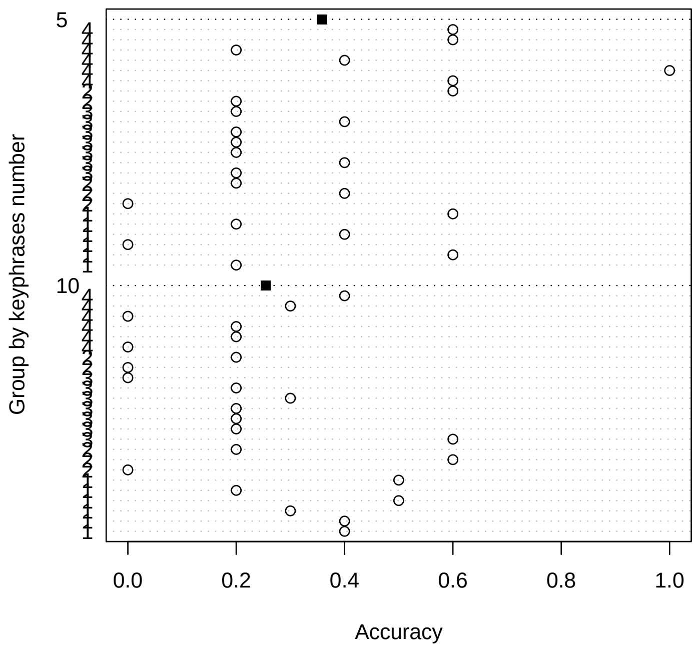


Figure 5.6: Overview dotchart 2: Accuracy Grouped by keyphrase Number

always factor level minus one; *Sum Sq* stands for sum of σ^2 which is the variance of the particular factor; *F value* is its estimated value in F distribution , and *Pr(≥F)* is the possibility of getting a value larger or equal to F than the estimated F value in F distribution. Usually *Pr(≥F)* is used to compare with α value which indicates type one error percentage.

From the above ANOVA table, we first examine the first hypothesis about the main effect of three factors. We can tell that after comparing with the other two factors that *klength*, denoting *keyphrase number*, has the smallest P value. This means it has great effect than the other two factors in influencing the final accuracy. However it isn't small enough to less than any reasonable chosen *alpha* value. We can write the formal hypothesis test as following, where a_i stands for the factor effect on the ith level of factor *keyphrase numbers(klength)*, b_i indicates the factor effect on the ith level of factor *domain(domain)*, and c_i denotes the factor effect on the ith level of factor *document size(tlevel)*. α here is set to be 0.05.

Null Hypothesis One: $H_0 : a_1 = a_2 = 0$

Alternative Hypothesis H_a : not all a_i is equal to zero.

Compare: $P[F_0 \geq 3.7227] \simeq 0.06287$ where $F_0 \sim F[1, 31]$ under H_0.

Decision: Since P value(Pr($\geq$)) is larger than $\alpha = 0.05$, we failed to reject the H_0.

Conclusion:There is not sufficient evidence to suggest that the keyphrase length has an effect on influencing the final result.

Similarity We can conclude that the factor *domain* and *document size* also have no effect in influencing the final result.

This conclusion is interesting and may be pragmatic because the factors, except the keyphrase number, are characteristics of documents; they should have no effect since they are hard to control in any algorithm. Otherwise our algorithm doesn't depend on domain-specific training sets to achieve highly-accurate results and also must not bound the size of the document. This is the reason that the other two factors also take no effect. Besides the above mentioned, the keyphrase number may have another story. From Fig 5.6, we can see that a smaller number of keyphrases would generate a more accurate result. This may be caused by the measurement method we used which is a ratio of the number of matches to the number of keyphrases. This also indicates that matches have no relationship to the number of keyphrases we set in the algorithm. This is also a feature of the heuristic rule approach.

Second Objective: Interaction Effect

From the ANOVA table, we can clearly see there is an interactivity effect existing between the number of keyphrases and the domain. The interaction effect between domain and document size is not obvious but close to critical region. Before we are going to perform formal statistical tests, we first glance at their interaction plots.

We can easily tell whether or not there is an interaction effect between two factors by looking at the parallel pattern between their tendency lines in the interaction plot. In the Fig 5.7 and Fig 5.9 we can see the dramatic differences in their tendencies. In Fig 5.8, there

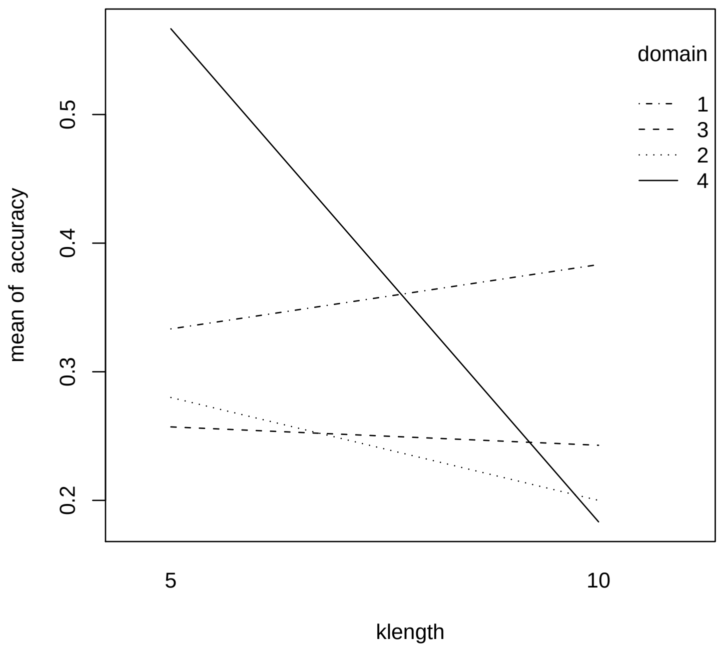


Figure 5.7: Interaction plot between keyphrase length and domain

is no parallelism. However, from the test, we can see the interaction effect is not obvious, indicated by its large P value. We can ascribe this lack of parallelism to the small sample data size.

The formal statistical analysis is given as follows, where ab_{ij} stands for the interaction effect between the ith level of factor, *keyphrase numbers(klength)*, and the jth level of factor *domain(domain)*. And ac_{ij} denotes the interaction effect between the ith level of factor, *keyphrase numbers(klength)*, and the jth level of factor *document size(tlevel)*. And bc_{ij} denotes the interaction effect between the ith level of factor, *domain(domain)*, and the jth level of *document size(tlevel)*. It is customary to set α at 0.05.

Null Hypothesis One: H_0 : All $ab_{ij} = 0$

Alternative Hypothesis H_a : not all ab_{ij} is equal to zero.

Compare: $P[F_0 \geq 3.1911] \simeq 0.03719$ where $F_0 \sim F[3, 31]$ under H_0.

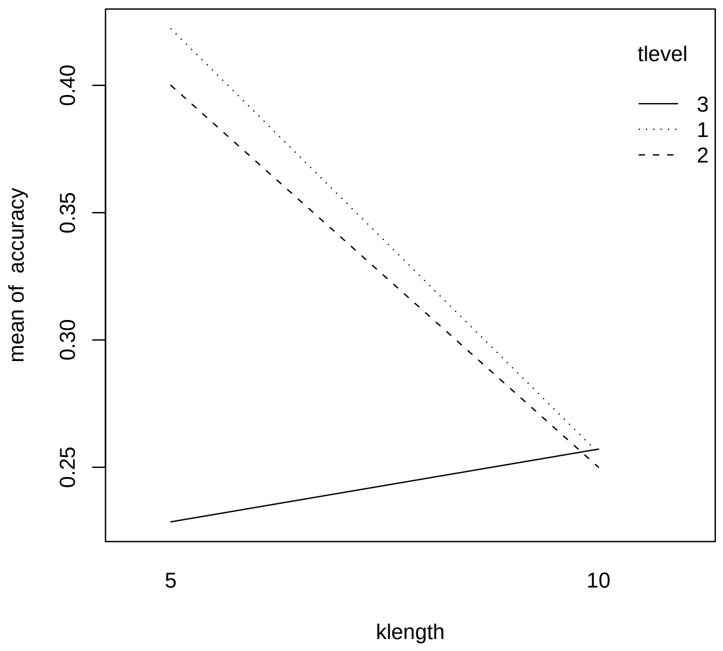


Figure 5.8: Interaction plot between keyphrase length and document size

Decision: Since $P[F_0 \geq 3.1911]$ is smaller than $\alpha = 0.05$, we reject the H_0.

Conclusion:There is sufficient evidence to suggest that there is an interaction effect existing between the keyphrase length and domain.

Null Hypothesis One: H_0 : All $bc_{ij} = 0$

Alternative Hypothesis H_a : not all bc_{ij} is equal to zero.

Compare: $P[F_0 \geq 2.4383] \simeq 0.05644$ where $F_0 \sim F[5, 31]$ under H_0.

Decision: Since $P[F_0 \geq 2.4383]$ is greater than $\alpha = 0.05$, we failed to reject the H_0.

Conclusion:There is no sufficient evidence to suggest that there is an interaction effect existing between the domain and document size.

Similarity we can conclude there is no obvious interaction effect between number of keyphrases and document size.

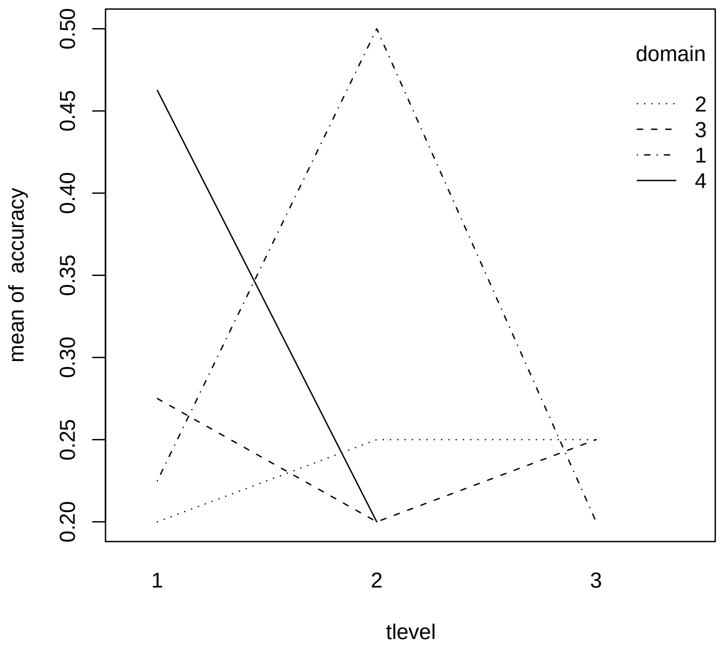


Figure 5.9: Interaction plot between documentsize and domain

This discovery is not out of our expectations since the domain and number of keyphrases should have a correlation that infuences the final result. This means that if we trained some well-defined data upon a specific domain, different levels of number of keyphrases would bring a difference in the final result. We may also draw the conclusion that the if we can combine the heuristic method and learning approach together, the accuracy would improve compared to before.

5.6.4 Study Validity Consideration

Validity threats are usually raised from several aspects: Conclusion Validity, Internal Validity, Construction Validity and External Validity. The possible factors that can trigger these threats are usually classified into several major perspectives such as time order, environment change, subject mortality, and others. For our study, we also have issues which might raise the threats of validity. We conclude them as follows:

1. Conclusion Validity

   Our conclusion is drawn by statistical analysis. This step should not have substantial problems. However due to the interaction effect existing between a block factor and a general factor, we should take careful consideration of the experimental study.

2. Internal Validity

   This validity related to whether or not the logic in this algorithm is correct. This step should be correct. In the algorithm, currently we pass in very few parameters. To improve the result quality and also eliminate the internal validity threats, we should specify the parameters as implicitly as possible in our future work.

3. Construct Validity

   This step has some validity threats. The largest threat is that the final number of matches number is tabulated by a human. Another threat is caused by our sample data size, which is too small. However the latter threat is largely due to resource limitation and cost issues. There are also some construction validity concerns caused by confounding factors since we don't examine all factors.

4. External Validity

   This study can be also conducted with other algorithms. Since it is involved with any the time issue so it also can be applied on others without consideration on time order.

5.7 Conclusion and Future Work

In this chapter, we introduced the VKE algorithm and its experimental result. The algorithm combines heuristic methods and a learning model. Experimental analysis is used to discover two things: (1)important factors which might impact the accuracy and (2) whether or not there is an interaction effect existing in factors. Our finding reveals that the number of keyphrases as a parameter in the algorithm has more of effect than other two factors, document size and domain specifics, in influencing the final result. Consequently, there is an interaction effect existing between the number of keyphrases and a specific domain. This experiment also has its disadvantages such as small sample size. In the future, we should focus on building up an empirical regression model which would serve to automatically analyze a factor's effect when a new factor is introduced into study. Also our VKE algorithm should be improved by combining training-purpose features and heuristic rules. We can accumulate data from a particular user as a training data set which is only specific for him/her in our algorithm. This would help us to gain a large performance promotion in anuser-based context awareness collaborative environment.

Chapter 6

Context Centric Model

In this chapter, I will illustrate how the user centric model is established upon the ontology pattern. The user centric model covers three parts: how we establish the user profile based on the ontology information, how the system can know the user current context and what the work flow of a typical knowledge discovery process is.

6.1 Acknowledge User

The basic knowledge unit in KAM, what we called JAN, can be abstracted from various source types. It can be extracted from a concrete object like a document, webpage or an email. It can also be a logic unit with references to others. Acting as the basic reusable resource unit in the ROA architecture, JAN plays a large role in representing knowledge. As mentioned earlier, it is a revision of the LOM standard. According to the definition of LOM, it should contain important fields such as annotations, references, and categories. These properties are essential in our KAM model when dealing with knowledge processing and discovery. From a user's perspective, the knowledge units known as JANs are unique to themselves, reflecting the user's interest and preferences. The annotations and references which the user manually labelled on the JAN emphasize the importance the JAN in user's view. More generally speaking, a JAN as a representation of knowledge should not only contain Meta information describing its conceptual meaning but should also contain meta information reflecting the view of a user. In another words, a JAN is only meaningful when it is combined with user information. As mentioned in chapter three, a standard user usually crosses over multiple domains. For instance, Professor Dr.Watson in chapter three's scenarios has a family domain besides his personal family domain. According to the KAM definition, a KAM resides on only one domain in which a user's ontology manages resources' classifications and their relationship. A particular user ontology maintains two parts: taxonomies containing JANs and taxonomy relationships. Considering this, a user profile in KAM should be augmented with a user's ontology.

6.2 User Profile based on Ontology

A typical user profile contains two parts. One is user basic information and the other is a user's preferences. The former part is unique for every user. The latter part plays an important role in the knowledge engineering model. In KAM , the latter part is described by a user's ontology, in which the taxonomies are user preferences. In addition to the typical user profile, in KAM, a set of rules is generated from a user's behavioral history. A typical user behavior describes the user's action upon a knowledge unit. An example is a user browsed a particular webpage or read a specific document. The user's behavioral history is the set of records of user behavior. By analysing the user's behavior and the knowledge units related to them, the KAM organizing agent can help the user to find related JANs more accurately from two aspects. (1). A new JAN classification: When a new information object comes in, the organizing agent(oA) in the Vijjana framework needs to find a suitable category in which it can reside. This is transformed into a query on the user profile and the query results containing all possible taxonomies. (2). Find related JANs: the content of the most browsed JANs are extracted and a query containing this content is submitted to the KAM discovery agent. The query would return the JANs with highest content similarity. Traditional information retrieval methods rely on keywords list to describe the content of an information unit. In KAM, the information unit, JAN, is organized into a taxonomy. 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 which returns the taxonomies in which the knowledge unit resides. To implement that, we establish several statistical probability models to map phrases into concepts. In the KAM model, the T set defines these concepts as taxonomies. For each JAN, its original resource content is extracted using Vijjana KeyPhraes extraction algorithm(VKE) and a similarity test is performed along all user taxonomies to determine its concepts. If there is no concept that is best fit, a universal similarity test is performed on a public ontology. Based on [?], approximately 3,000 terms will cover all general concepts for a specific domain. For a user, It is possible that we can use a finite taxonomy to cover all domains he/she crosses over.

6.2.1 Global ontology

To construct a user profile with ontology information, we need a large reference data repository. Open Directory Project (ODP) became to be the final choice after we reviewed several data sources. It is well accepted and has been cited in several papers [?][?]. The ODP is regarded as one of the largest taxonomy stores for web directories. The taxonomy is organized with a hierarchical structure. In [?] paper, they also used ODP as their main reference source and concluded that the using the top three levels of taxonomy as references would promote the ontology hit accuracy. In the KAM framework, we also follow this suggestion and use taxonomies in the first three levels as our global concept set. Our purpose is to construct a universal ontology. We first analyse 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 need to reorganize them into one unit. In KAM, the ontology is defined as a set of taxonomies and each taxonomy contains knowledge units related to it. For a taxonomy in the KAM universal ontology, we also expect it to have these two features. After investigating the ODP data, we found it also has two similar features. (1). It has siblings on the same level. (2). For every node in the hierarchy, we can find corresponding items in the RDF file. The RDF item is usually a bookmark link and its self-description. So here we can map the ODP hireachy node as our universal ontology taxonomy and the RDF item as the knowledge unit JAN in the universal ontology. So a universal ontology taxonomy defined in KAM has relationships between its parents and siblings that are also taxonomies. It contains knowledge units, the links defined in the RDF. We can rely upon these features to construct our universal ontology. The Fig. 6.1 is a partial universal ontology view. The first level contains 14 taxonomies. The second level contains 517 taxonomies. And the third level contains 6056 taxonomies.

6.2.2 Constructing user ontology

In KAM, the user ontology manages all user knowledge units. The knowledge units mainly consist of three parts: User Email, User File System and User bookmarks. Here we only illustrate the methodology used in constructing a user's ontology from the User FileSystem and User bookmarks.

JAN Abstraction

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

- extract content.
- PREPROCESSING: remove stop words, stemming.
- FULL TEXT or VKE processing.

The first step is extracting content. For the User file system, the current implementation supports three basic types of textual documents: text, pdf and MS Office Word. For bookmarks, we extract the content of bookmarked webpage. Secondly, In the stopword removal process, we search a large stopword corpus and remove any words in the document that appear in the corpus. The part remaining after this is passed directly to Stemming process which removes token suffixes and recovers the base, or stem, of the word. Step three has two options: FULL TEXT or VKE processing. The FULL TEXT processing uses all results from step two for keyphrase indexing. The VKE processing only builds an index upon the

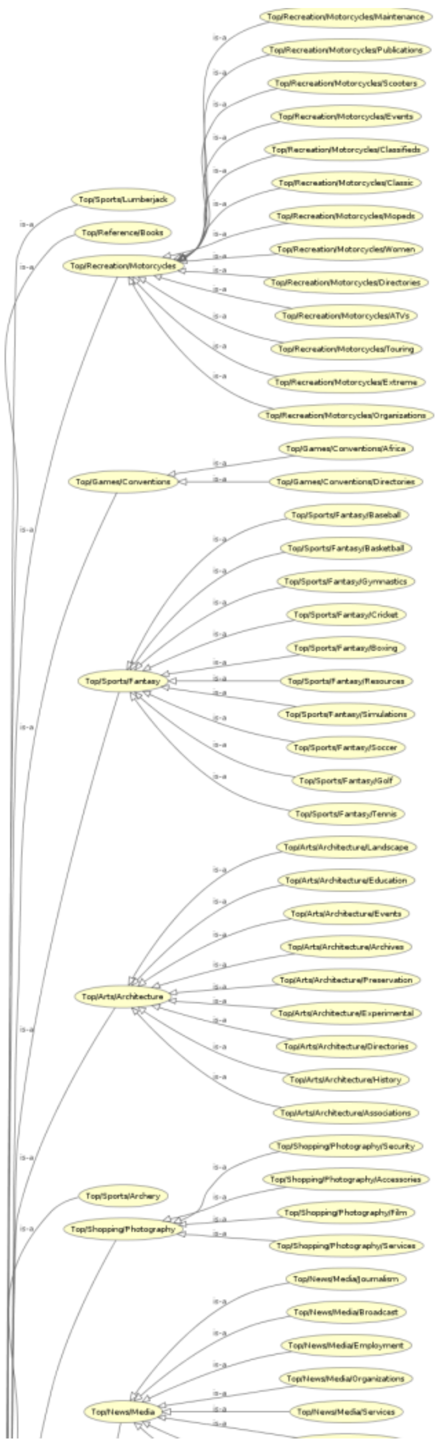

Top/Recreation/Motorcycles/Maintenance
Top/Recreation/Motorcycles/Publications
Top/Recreation/Motorcycles/Scooters
Top/Recreation/Motorcycles/Events
Top/Recreation/Motorcycles/Classifieds
Top/Recreation/Motorcycles/Classic
Top/Recreation/Motorcycles/Mopeds
Top/Recreation/Motorcycles/Women
Top/Recreation/Motorcycles/Directories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

Figure 6.1: Global Ontology

keyphrases generated by VKE algorithm. The phrases used in an index would are considered as annotations of this JAN. After the three steps of abstraction, a JAN is created.

6.2.3 User File System

To build a user ontology from the user's File System, we need to traverse through all user folders and files contained in them. Because the folders have a hierarchical structure, we map them to taxonomies in the user's ontology. Correspondingly, files are regarded as knowledge units under the taxonomy. The real implementation in KAM is achieved by a further abstraction.

We implement two interfaces: "KAM FileSystem" and "FileDescriptor". The "KAM FileSystem" (KFS) manages the folder structure. To create a KFS, the users must specify the KFS root path. The KFS has the capacity to navigat through its subfolders. It can recursive deeply, to the lowest level, and report whole structure. During this process, it can call a "FileDescriptor" which is responsible for generating file meta information and extracting file content. The file content and file information is used by the JAN creation process. The FULL TEXT processing in KFS has two opinions: lucence indexing [?] or our own method which will be explained soon. A generated User ontology upon file system is shown in the following Fig.6.2

6.2.4 User Bookmark System

The user bookmark system is a web interface that allows users to save their own bookmarks. In Paper [?] we ilwlustrated our old Firefox plugin to help the user through the JAN creation progress. The process is simplified in our new web interface. In addition to that, from the web interface, a user can create a user space to store all his bookmarks. Also, this user space can be shared with the KAM FileSystem. 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 content from a linked source. The content will go through the same process as JAN abstraction. However from web service, FULL TEXT is only supported in

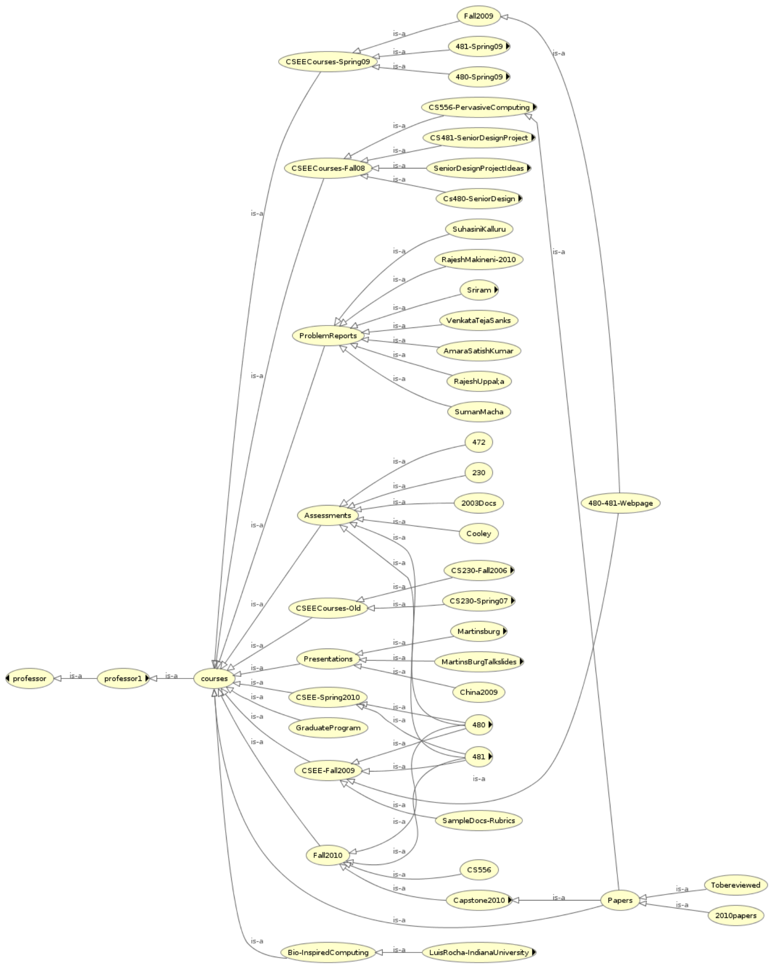
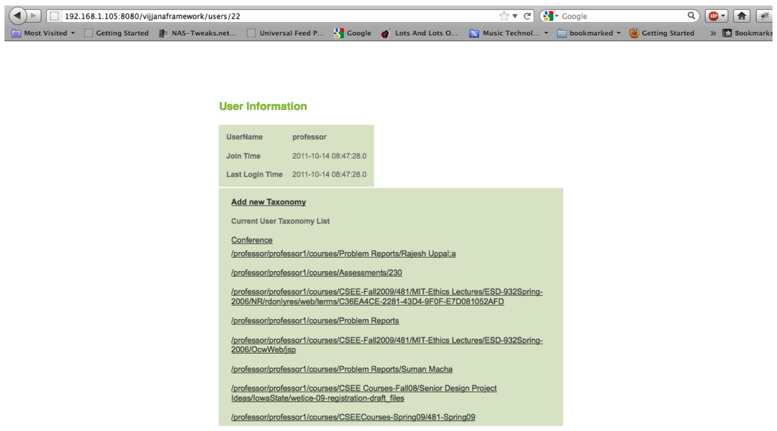


Figure 6.2: User Ontology Based on the FileSystem

our own version now. A ontology of the web interface is shown in the following Fig.6.3

6.3 Context Awareness

Another important part of a user's profile is the user behavioral model. Our premise 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 reveal the user preferences, we can generalize a finite number of rules by monitoring transitions happening between taxonomies. For instance, when off of work, a user who has a strong interest in sports may spend more time reading sports news


192.168.1.105:8080/vijjanaframework/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 Uppal;a

/professor/professor1/courses/Assessments/230

/professor/professor1/courses/CSEE-Fall2009/481/MIT-Ethics Lectures/ESD-932Spring-2006/NR/rdonlyres/web/terms/C36EA4CE-2281-43D4-9F0F-E7D081052AFD

/professor/professor1/courses/Problem Reports

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

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

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

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

Figure 6.3: User Ontology on Web Interface

than reading financial news. For this particular use, when the transition from work to news occurs, KAM shall promote the sports new 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 prior is the number of times the user browsed the taxonomy and the later denotes how many JANs are related to this taxonomy. We keep updating the taxonomy interest score when user browses the taxonomy or adds new JAN into it. The interest score is calculated as

$$I_{t_i} = \frac{total\_hit\_number}{taxonomy\_size}, Where \tag{6.1}$$

The I_{t_i} stands for the user interest on taxonomy i, *taxonomy_size* is the number of JANs in this taxonomy.

As we illustrated already, a user behavior models our base for context awareness. To detect in which context a user resides, KAM uses methods that fall into two categories: timeline and knowledge hit statistics. Before proceeding to explain our context awareness model, we need first 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 a wonderful dinner. When a person reads a book, the context is the paragraph that is engrossing him. In a knowledge network, for example the ODP project, the context is the branch where people click links. In the KAM model, we define the context as the current ontology on which the user is working. Recall in the user behavior model, a transition is 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}}.

6.3.1 Timeline Context Awareness

In the KAM web interface, every operation upon a taxonomy and JAN is recorded as user history. When a new taxonomy or JAN is created, its creation time is logged. In addition to that, We also keep a record when he 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 of everyday, always browses a certain taxonomy or related JANs, KAM would mark this time period with this taxonomy information and correspondingly set it as user context for this time slot.

Using this method, we first section the day off as being part of one of two classifications; the user is either active or inactive. The basic time unit can be one hour. The inactive periods are time units without any user activity. In opposition, the user has activities during the active period. For a certain time phrase of enough length, like one week or month, the user's activity can be categorized by these two periods. For the user's active period, we can detect the taxonomy boundary 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 phase, $P(t_i|time_j)$, and then choose the highest one as the user's context. We can see this from Fig. 6.4

6.3.2 Interest driven context

The another context awareness model concerns the taxonomy interest I_{t_i}. For each taxonomy, it has a interest score calculated as explained in the user behavior model. For all

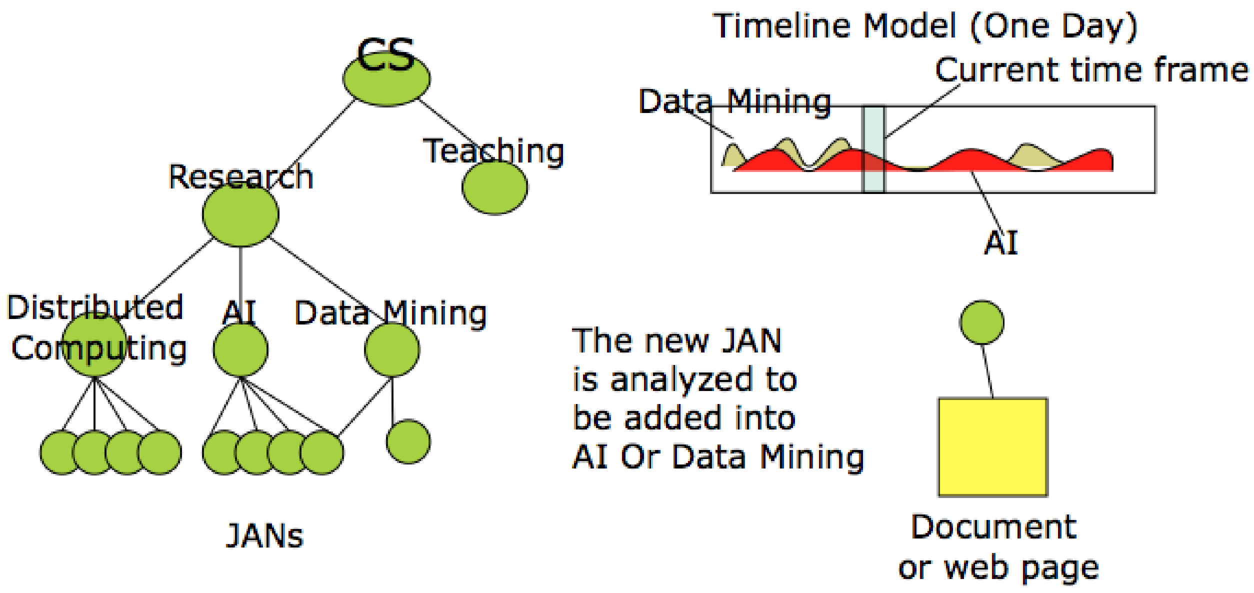


Figure 6.4: **timelinemodel**

taxonomies, statistically, each concept has a factorial value between zero to one to describe its importance.

$$PI_{t_i} = \frac{I_{t_i}}{\sum I_{t_i}} \tag{6.2}$$

By this weight value, we also can predict the most probable next state. This memoryless 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 given from a transition matrix. It is more likely that a user would move from the current state to the 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. After the context switch, every transition would update the interest score and consequentially update PI_{t_i}. Generally for a small user knowledge network with a low average hit number, the update operation would not be costly.

We can use Fig.6.5 to illustrate the Interest Driven Context Model.

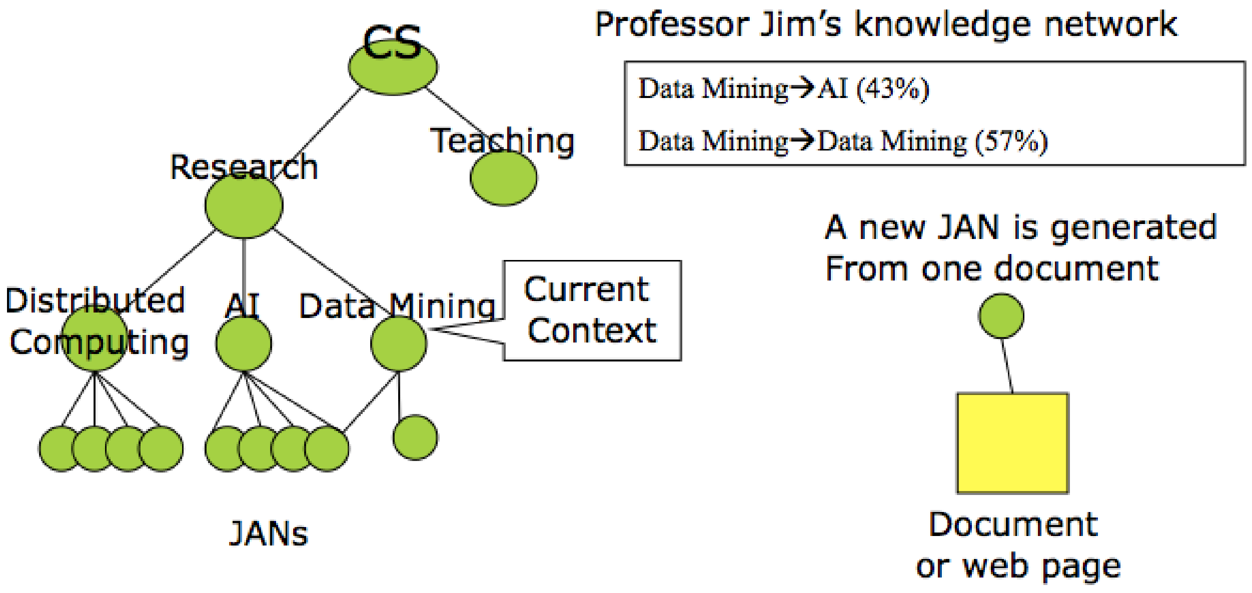


Figure 6.5: **Interest driven model**

6.4 Classification of JAN

Till this point, KAM has enough information to provide user a suggestion on how to organize their knowledge unit, JAN. KAM shares a universal ontology build from ODP data. Additionally, KAM owns its own user ontology and user behaviour history. All these would be used when classifying a JAN.

As we mentioned already, after the JAN abstraction progress, JAN annotations are also created and would be used as index phrases. Therefore a JAN is 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 knowledge units are all 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 user. When a new JAN is added into a user's ontology, the methodology of selecting a suitable taxonomy used for both of the global and personal ontologies is different. However, there is a commonality in training data processing of global and personal ontology.

6.4.1 Generate training date

As mentioned earlier, TF*IDF is a popular technology used in text classification. We also use it to do our basic classification. The detailed process is described as following:

$$w_{ij} = tf_{ij} * idf_i, \text{ where} \tag{6.3}$$

$$tf_{ij} = \text{term weight} \tag{6.4}$$

$$idf_i = \log \frac{\text{JAN training set size}}{\text{number of JAN containing } t_i} \tag{6.5}$$

Different to our VKE algorithm, here the term weight of a JAN annotation is calculated using the augmented normalized term frequency. The augmented normalized term frequency is described as:

$$AN(tf_w) = 0.5 + 0.5 * tf_w/tf_{max} \tag{6.6}$$

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 dependence of classification results on annotations length[?]. 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 need to specify the training set. In the global ontology, we use the top three taxonomy levels as a training set. In the user space, we employ all user created taxonomy data as training set. Therefore a taxonomy word vector consists of the sum up of all its containing knowledge units JANs' word vectors. We use V_t denoting a taxonomy word vector and j_t denoting JAN word vector. $V_T = \{V_j|\text{for all j in T}\}$.

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

$$TFIDF_{jan} = \sum w_i|\text{where}w_i = tf_i * idf \tag{6.7}$$

Consequently, the taxonomy weight is described as:

$$TFIDF_{Taxonomy} = \sum JAN_i | \text{where jan belongs to taxonomy.} \tag{6.8}$$

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

$$CN(V) = (w_1^*, w_2^*, \ldots, w_n^*)\, where \tag{6.9}$$

$$w_i^* = \frac{w_i}{\sqrt[2]{\sum w_i}} \tag{6.10}$$

6.4.2 Similarity between JAN and taxonomy

In order to determine a suitable taxonomy for an incoming JAN, we use cosine similarity to give 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 large to a user taxonomy vector. Let $T = \{w_0, w_1, \ldots, w_m\}$ represent the taxonomy and w_j stand for the term that appeared in it. For these two vectors, we form their own weight vector using the term's TFIDF value. To calculate their cosine similarity, the two vectors' lengths must be equal. Then we need to construct two equal length vectors. We first merge the two word vectors together to form a new vector, then we use this method to construct the weight vector: for every word missing in the original word vector, we fill its weight with 0. After finishing this, we can calculate the cosine similarity.

The cosine similarity of these two vectors can be expressed as :

$$cosine(t_i, JAN_j) = \sum w_{ik} * w_{jk}, Where \tag{6.11}$$

$$w_{ik} \text{ indicates the TFIDF weight of term k appears in the taxonomy i} \tag{6.12}$$

$$w_{jk} \text{ indicates the TFIDF weight of term k appears in the new JAN j} \tag{6.13}$$

$$\tag{6.14}$$

The final cosine similarity value is between -1 and 1. The close the absolute value is to 1, the more similar the two vectors are. Consequently we can see the JAN is similar to the taxonomy. The taxonomy becomes a candidate. In addition to this value, we also consider the Interest hit value. In the final suggestion rank, weight consists of two parts: the cosine similarity result and PI_{t_i}.

$$R_T = Cosine(T, j) * PI_{t_i} \tag{6.15}$$

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

6.4.3 Relationship between local taxonomy and global taxonomy

In KAM, one important part is the communication capacity between different users. A user 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, 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.

6.5 Knowledge discovery process

One user, at a given point in time, should only reside in only one context. As we defined above, the context provides the current taxonomy and next possible 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 concerning computer science. All communities share the universal ontology. So we can use a diagram, Fig. 6.6 to show a better understanding of this.

This also leads to our knowledge discovery process, which is a three-step procedure. We name this procedure “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 it then explores the universal cloud. Knowledge discovery can be initiated by a user in a certain context or an agent during context switching. No matter in which way this occurs, it is performed by queries upon a taxonomy. A typical query is constituted by a set of phrases.

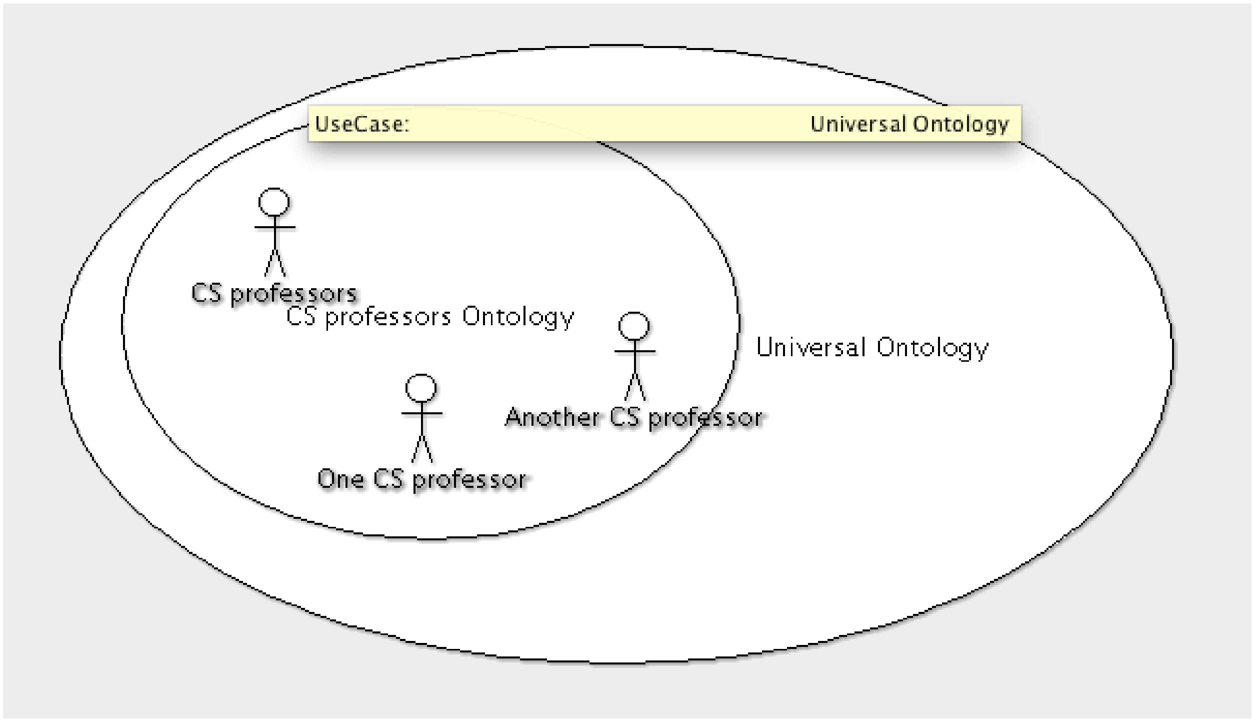


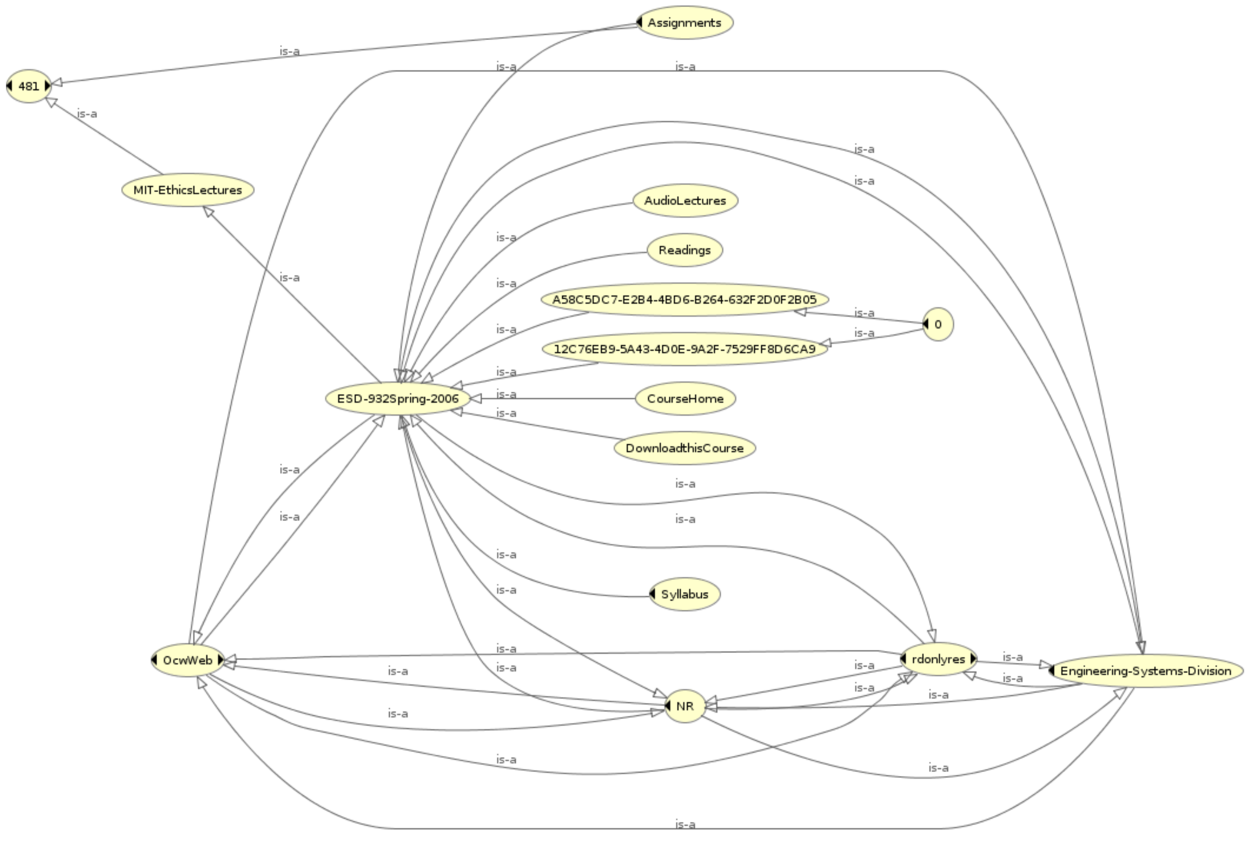
Figure 6.6: **A general view of ontology** - with annotations, categories and references fileds

6.5.1 Knowledge discovery process in local

The local search is confined in user taxonomies. For a given query, the cosine similarity described earlier can still be used. If the search is confined into a particular taxonomy range, for every JAN, we perform a similarity test and the JAN corresponding to the largest value is returned as the query's result. However if the search doesn't confine into any taxonomy, the similarity test is performed between 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 enacted upon the returned taxonomies is similar to what we did for an individual taxonomy.

Figure 6.7 describes a local search belonging to a professor. When he is preparing his course 481, he found part of his ontology on MIT courses he saved before.



s

Figure 6.7: Local Search

6.5.2 Knowledge discovery process in community

Since people in the same community share the same ontology, we can use the collaborative filtering (CF) technique to recommend JANs to a user. There are two type of collaborative filtering, user based CF and item based CF. Recall in the KAM model, in order to eliminate the problem for organizing and consistency checks, we apply the ROA architecture which requires that all items be uniquely identified. For an original resource, it is abstracted in to a JAN to be added into user's knowledge network, which is unique in the whole knowledge network. So here, we can't directly apply the CF technology upon JANs. There are two methods to solve this problem. First is the rudimentary one, to use the JAN's reference, the original resource, as our item. The other is using 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 is the hit number of the JAN. So Here the

$$UI_{ij} = \begin{cases} hitnumber_i & \text{if item i also appears in } user_j\text{'s knowledge network} \\ 0 & \text{Otherwise} \end{cases} \tag{6.16}$$

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. The performance between the item-based and user-based methods depends on the sparsity of the matrix we build. If the matrix is sparse, the user-based performance should be poorer than the item-based.

For the second method, we use a keyword to replace the JAN, so the the user-item matrix is formed in the following manner. Here the keyword stands as a set of JANs which use this keyword as index.

$$UI_{ij} = \begin{cases} 1 & \text{if keyword i also appears in } user_j\text{'s knowledge network} \\ 0 & \text{Otherwise} \end{cases} \tag{6.17}$$

We calculate the similarity between the two keywords and recommend JANs, sorted by the highest similarity keyword to the user.

6.5.3 Knowledge discovery process in universal

The final step in "Call it Once" is to search in the global ontologies. These relationships are defined in the universal ontology and the search is already out of the user's context. So here the query is without any user preference. For each related taxonomies from global ontology, we perform a local search on user's related taxonomies and regard returned JANs as a compensation of result from global search and community search. So the overall discovery process can be viewed as Fig. 6.8

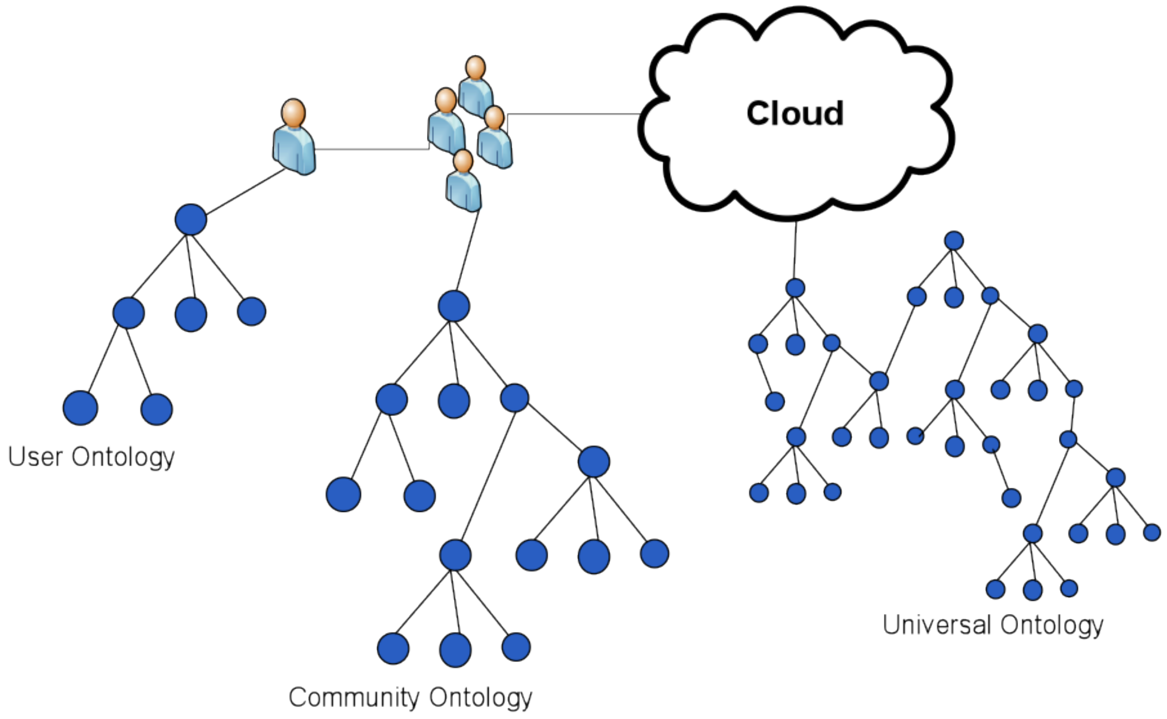


Figure 6.8: **The whole discovery process**

Chapter 7

Implementation and Applied methods

In this chapter, I will go through the implementation details of KAM. It illustrates three agents in KAM: discovery agent, search agent, and context agent.

7.1 KAM implementation Architecture

As introduced in Chapter three, KAM consists of three important knowledge bases and nine intelligent agents. All these agents work upon the three knowledge bases, and the processing results reflect back the knowledge bases. In order to fulfill this requirement, we design the KAM architecture as follows Fig. 7.1

As we can see from Fig 7.1, The top layer is the knowledge representation layer. It has two parts, a web interface and Personal File System. The web user interface is aimed to cover two knowledge bases: Personal bookmarks and Personal Email. This information is gathered and collected independent of physical media and can be accessed anywhere. The personal File System covers the knowledge base when stored in a physical media and with resource access restrictions. The second layer is the intelligent agent layer. Each agent running on this player acts in two parts. It can serve as a web service for the KAM web interface and also can execute as a standalone program while coping with a personal file system. The third layer is the logic layer taking charge of managing the conversion from the logic objects to data objects. It also acts as a channel for saving and retrieving data and their logic relationships back and forth between the KAM agents and the Database layer which is the fourth layer.

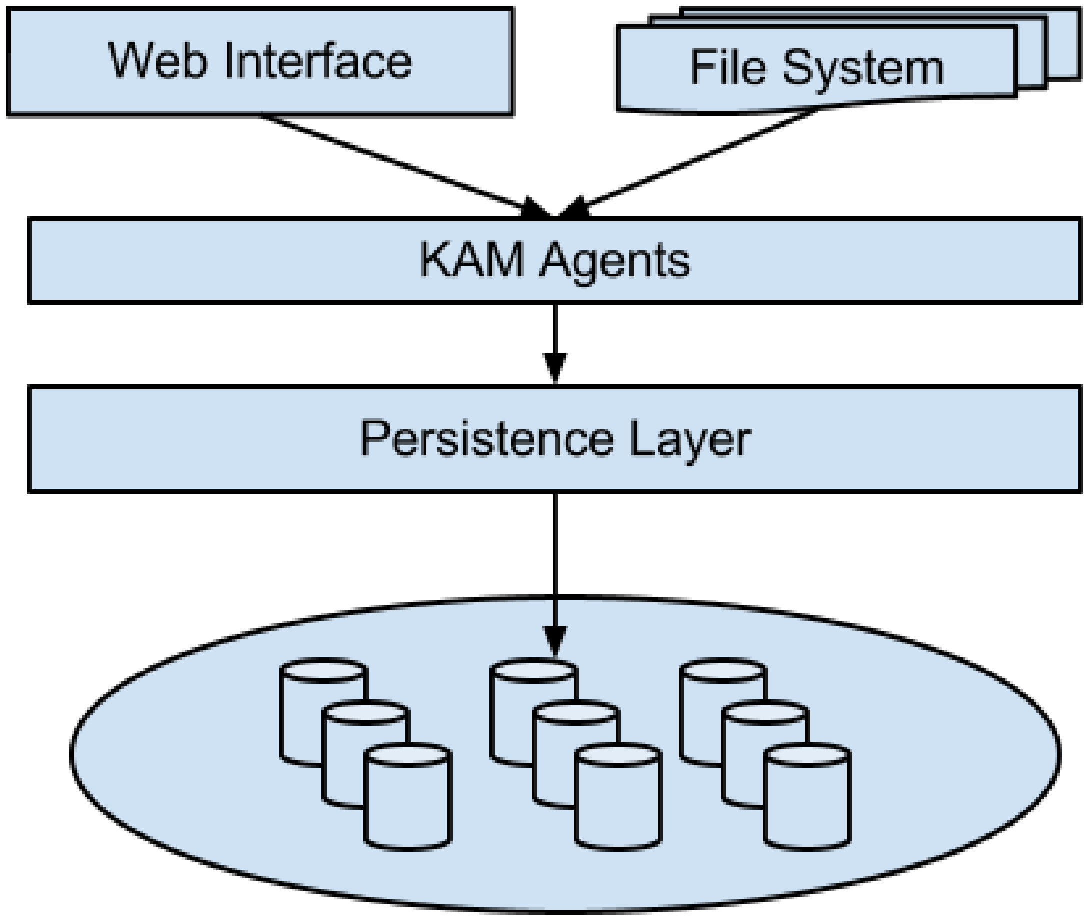
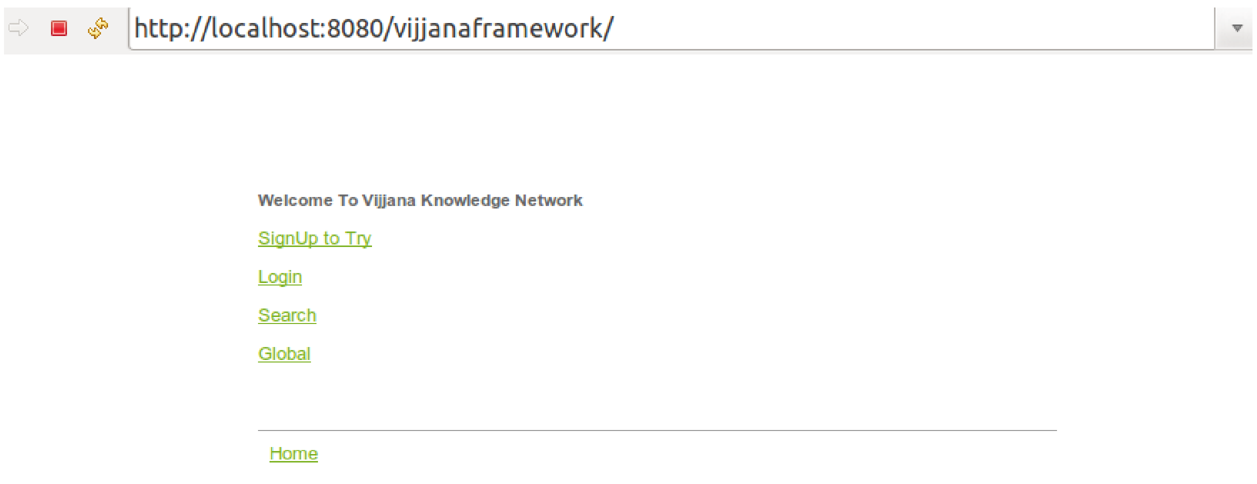


Figure 7.1: KAM Architecture

7.1.1 Knowledge representation layer

The web interface and file system comprise the knowledge network representation layer. The web interface provides a series of cloud-based web services to the user and also it is an ontology repository for user's ontology and global ontology. Fig.7.2 is the screen shot for the KAM entry point. From there, an anonymous user can register new users or browse the global ontology repository. The KAM web interface automatically creates a user space for each registered user that allows each user to save their personal ontology. The web interface also provides an interactive interface to registered user for all available KAM agents.


http://localhost:8080/vijjanaframework/

Welcome To Vijjana Knowledge Network

SignUp to Try

Login

Search

Global

Home

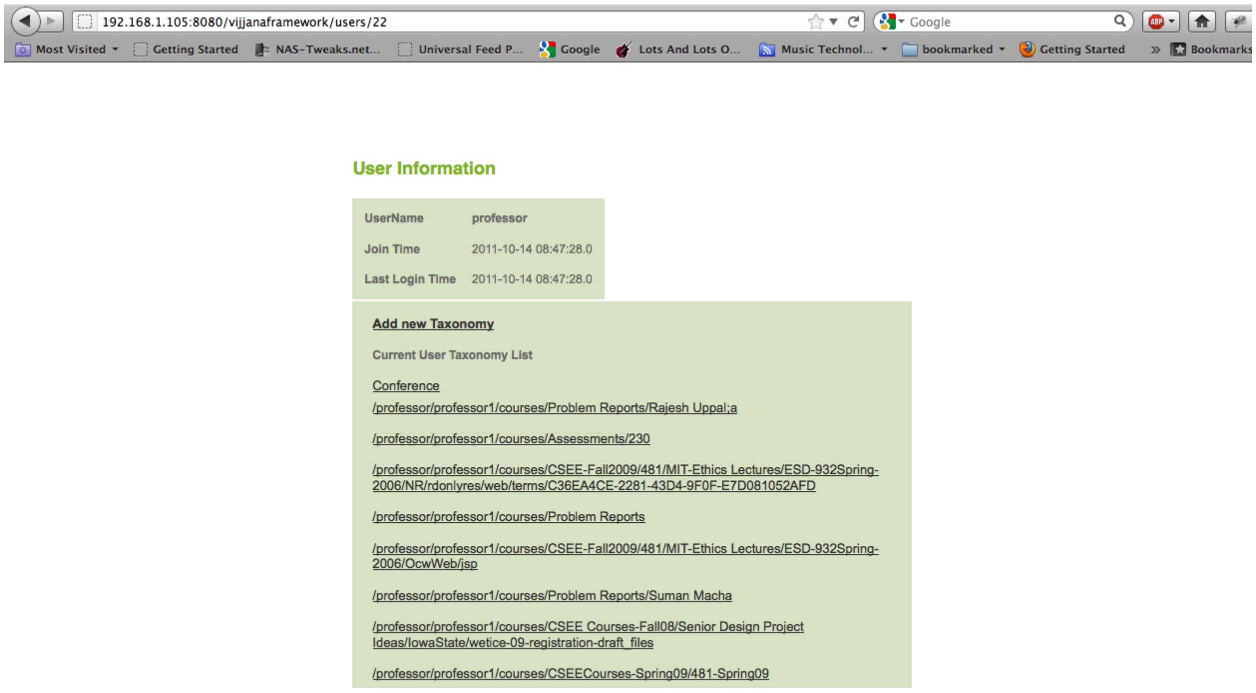
Figure 7.2: KAM Web interface entry.

The KAM agents were implemented as web services that can be accessed from everywhere. The already implemented web services are :

- User space on cloud.
- Search Agent
- Discovery Agent
- Markup Agent
- Visualization Agent (partially)

User Space

The user space is a user's online knowledge repository. Users can create taxonomies and JANs. Users can also link their personal file system with the web interface by displaying metadata of file item inside the personal file system. Fig. 7.3 is the example of a professor's ontology containing all web-created taxonomies and taxonomies created from the personal file system.



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 Uppal;a

/professor/professor1/courses/Assessments/230

/professor/professor1/courses/CSEE-Fall2009/481/MIT-Ethics Lectures/ESD-932Spring-2006/NR/rdonlyres/web/terms/C36EA4CE-2281-43D4-9F0F-E7D081052AFD

/professor/professor1/courses/Problem Reports

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

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

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

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

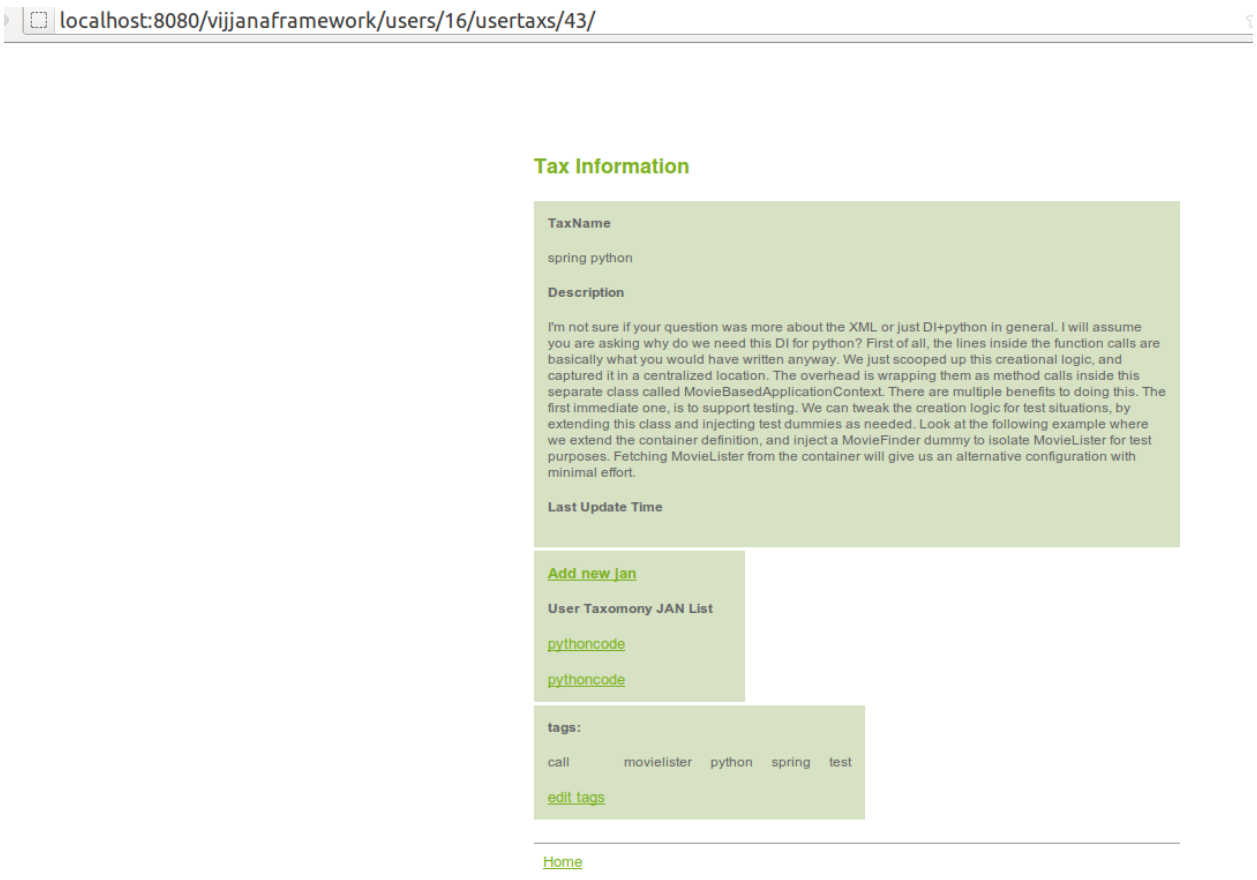
Figure 7.3: KAM User Space

Markup Agent

The taxonomy and JAN created from the web interface are automatically analyzed by the Markup agent. It generates annotations for taxonomies and JANs. The generation progress is illustrated in Chapter four. All these annotations can be edited in that the user adds and removes them through web interface. Fig. 7.4 shows a new taxonomy. The keyword is automatically generated and listed below the taxonomy description.

Discover Agent an Search Agent

The Search functionality in KAM is provided by the Discover and Search agents. The discover agent performs a "Call in Once" query process to discover related taxonomies and JANs when a particular JAN is given. The detailed process was explained in the prior


localhost:8080/vijjanaframework/users/16/usertaxs/43/

Tax Information

TaxName

spring python

Description

I'm not sure if your question was more about the XML or just DI+python in general. I will assume you are asking why do we need this DI for python? First of all, the lines inside the function calls are basically what you would have written anyway. We just scooped up this creational logic, and captured it in a centralized location. The overhead is wrapping them as method calls inside this separate class called MovieBasedApplicationContext. There are multiple benefits to doing this. The first immediate one, is to support testing. We can tweak the creation logic for test situations, by extending this class and injecting test dummies as needed. Look at the following example where we extend the container definition, and inject a MovieFinder dummy to isolate MovieLister for test purposes. Fetching MovieLister from the container will give us an alternative configuration with minimal effort.

Last Update Time

Add new Jan

User Taxomony JAN List

pythoncode

pythoncode

tags:

call movielister python spring test

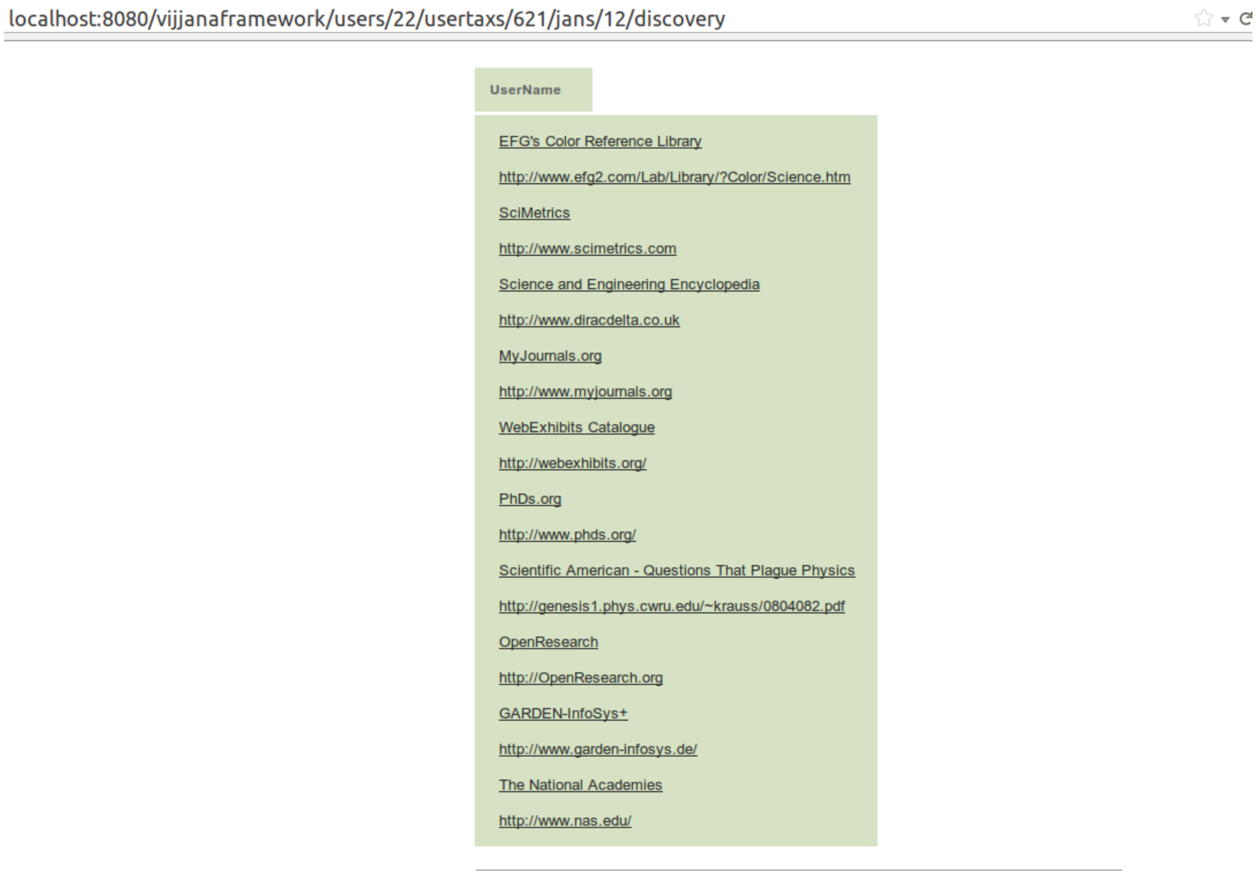
edit tags

Home

Figure 7.4: A new created taxonomy

chapter. Figure 7.5 is a screenshot showing the discovery agent found for user 22 who owns the JAN with id 12 under his 621th taxonomy.

Different than the Discover agent, the Search agent doesn't associate with a particular JAN. Users can issue any query keywords as one search that is parsed and regarded as special to a JAN. Correspondingly, the search keywords are regarded as its annotations. For an already logged-in user, the next process is similar to a "Call it once" process in discovery agent. For an anynomous user, the search process operates in the global ontology repositories and returns taxonomies and JANs which has the highest similarity with the search object.


localhost:8080/vijjanaframework/users/22/usertaxs/621/jans/12/discovery

UserName

EFG's Color Reference Library
http://www.efg2.com/Lab/Library/?Color/Science.htm
SciMetrics
http://www.scimetrics.com
Science and Engineering Encyclopedia
http://www.diracdelta.co.uk
MyJournals.org
http://www.myjournals.org
WebExhibits Catalogue
http://webexhibits.org/
PhDs.org
http://www.phds.org/
Scientific American - Questions That Plague Physics
http://genesis1.phys.cwru.edu/~krauss/0804082.pdf
OpenResearch
http://OpenResearch.org
GARDEN-InfoSys+
http://www.garden-infosys.de/
The National Academies
http://www.nas.edu/

Figure 7.5: Discovery agent found related JAN

7.1.2 Visualization Agent

The visualization agent provides the visualizable data to the front end. Currently, it generates the graphML format data from the user's ontology structure. This graphML originally was read in our original visualization tool developed upon Prefuse[**?**]. Besides that it has been concluded that Prefuse is difficult to integrate into our framework, we are still investigating some better graphic toolkits like JUNG. Fig. 7.7 shows an example of graphML that we generated from a particular user's ontology. The nodes in this graphML are user taxonomies.

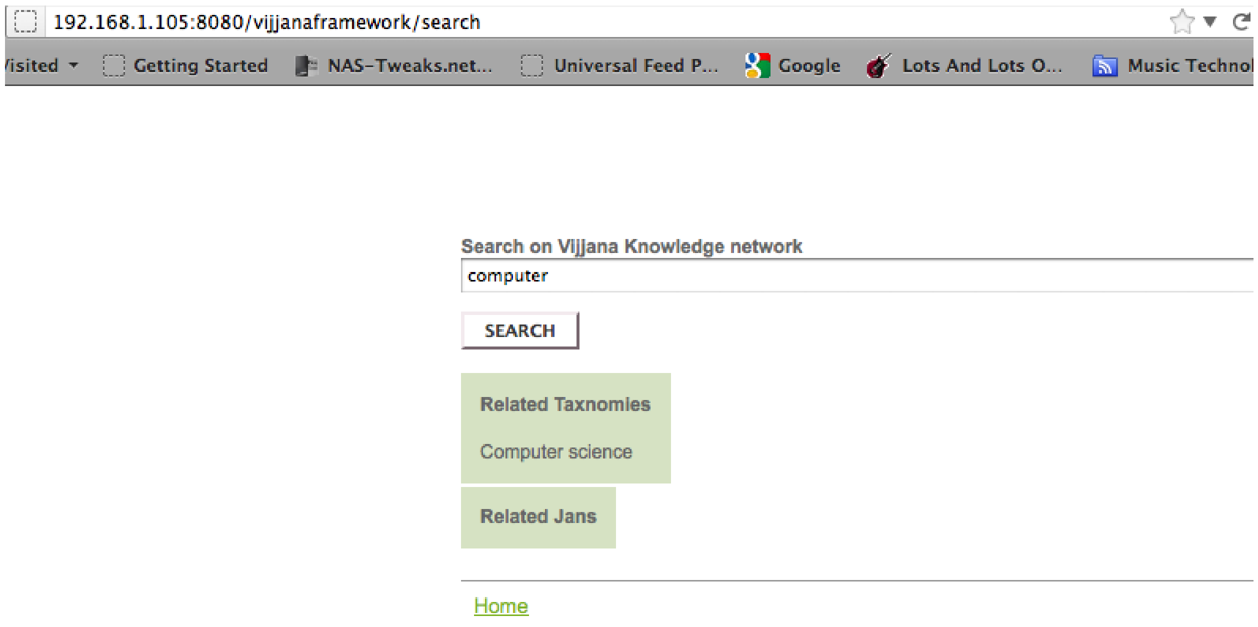
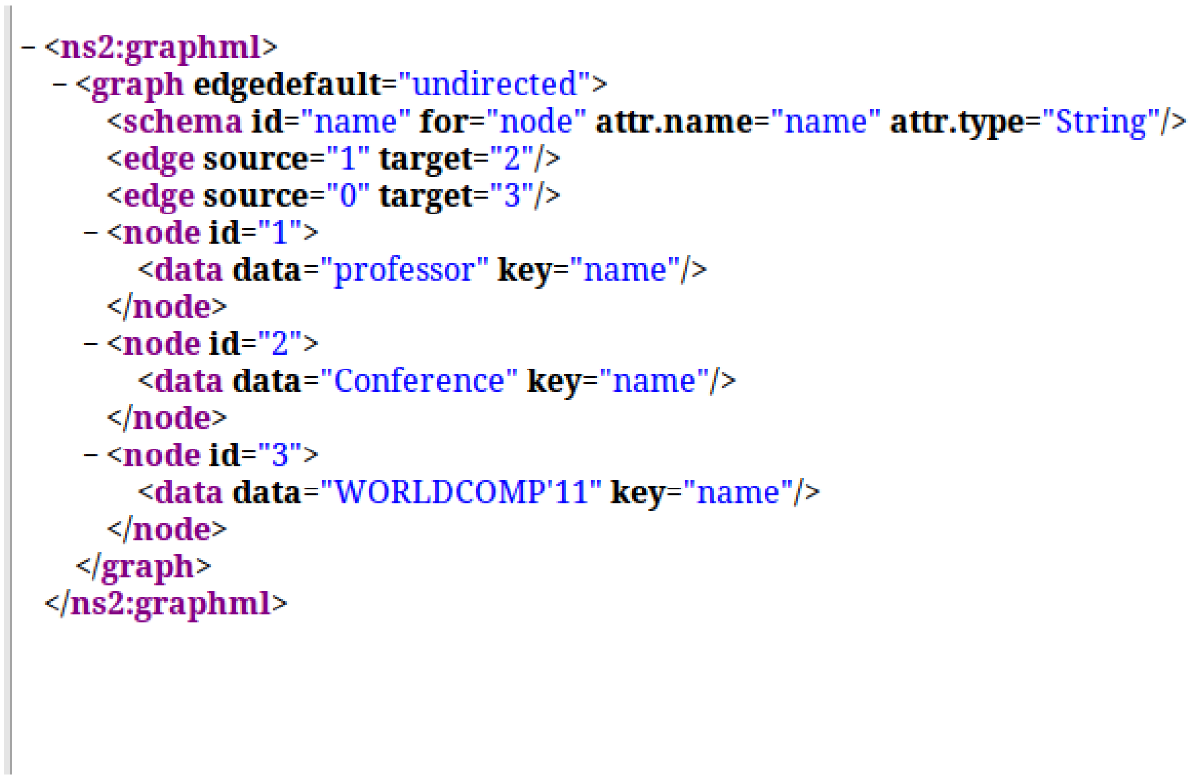


Figure 7.6: Search agent return related global taxonomy

7.1.3 Implementation technology

There are plenty of mature web frameworks available on market for use. Eventually We chose the Spring framework as our servlet container. It inherits all the J2EE application server's features but is lightweight on servlet deployment and object injection. Meanwhile, the MVC model inside Spring offers us the ability to dynamically map all agents running results directly to a web view. This gives us more flexibility on the knowledge representation layer. We could alter our data model with more enriched display elements. Along with all above, the newest version of Spring framework fundamentally supports all RESTful operations, which eliminates the barriers of communications between different agents.



```
- <ns2:graphml>
  - <graph edgedefault="undirected">
      <schema id="name" for="node" attr.name="name" attr.type="String"/>
      <edge source="1" target="2"/>
      <edge source="0" target="3"/>
    - <node id="1">
        <data data="professor" key="name"/>
      </node>
    - <node id="2">
        <data data="Conference" key="name"/>
      </node>
    - <node id="3">
        <data data="WORLDCOMP'11" key="name"/>
      </node>
    </graph>
  </ns2:graphml>
```

Figure 7.7: Sample graphML

7.2 Middle Layer, Database Layer and Development Environment

7.2.1 Middle Layer

The middle layer plays an essential role in the whole KAM architecture. It is the bridge connecting the upper agent layer and the lower data warehouse layer. The KAM clients running on users' desktops store data into the same database that the web interface accesses. In the real implementation, we use Oracle JBOSS Hibernate as our object relationship management tool. From a programming perspective, the interface provided to the web interface and file system is the same. However the details in implementation are different with a variation of different representation layer requirements. For the file system, we need to manually manage database session creation, transaction initialization, and commit. For the web interface, all these are handled by Spring[?]. When the Spring framework instance starts, it creates a Hibernate session factory[?]; the session factory is always kept running until the Spring instance is shut down.

Aside from the Hibernate layer, KAM also contains a cache layer. There are two types of cache data stored in this layer. One stores the ODP data and the other caches the user ontology data. All these are cached into files. For the standalone application, this data is loaded on demand. In the web interface, the data is loaded when application server starts and stays in memory as the application server runs. KAM also provides the functionality to serialize the cache data. The cache data can be serialized into files that can be distributed to other KAM instances. It provides a way for importing and exporting data.

7.2.2 Database Layer

KAM right now runs on a relational database containing twenty tables. There are five important tables in two categories: (1) UTAX and UAN are two tables storing the global ontology data distilled from ODP data set. (2) vijusertax, vijuserjan and vijtag are three tables storing the user ontology data. The relationship of user taxonomies and JANs, user taxonomies and global taxonomy, user JAN and its annotations, user taxonomy and its annotations are described by four join tables, eg. vijuser_has_tag. Figure 7.8 shows the database tables.

We use MySQL as our backbone database. It is running on a dedicated database server.

7.2.3 Development tools and source architecture

The current KAM is implemented purely in Java. The whole KAM source contains three namespaces and twenty-four java packages. The namespace “com.vijjana.framework” is the main namespace of the KAM framework. All the intelligent agents are implemented under this namespace. The “com.vijjana.keyphrase” namespace is designated for the VKE algorithm that is illustrated in Chapter 5. The “com.vijjana.util” namespace contains all utility classes that are used in KAM. We use maven for our package management, compile,

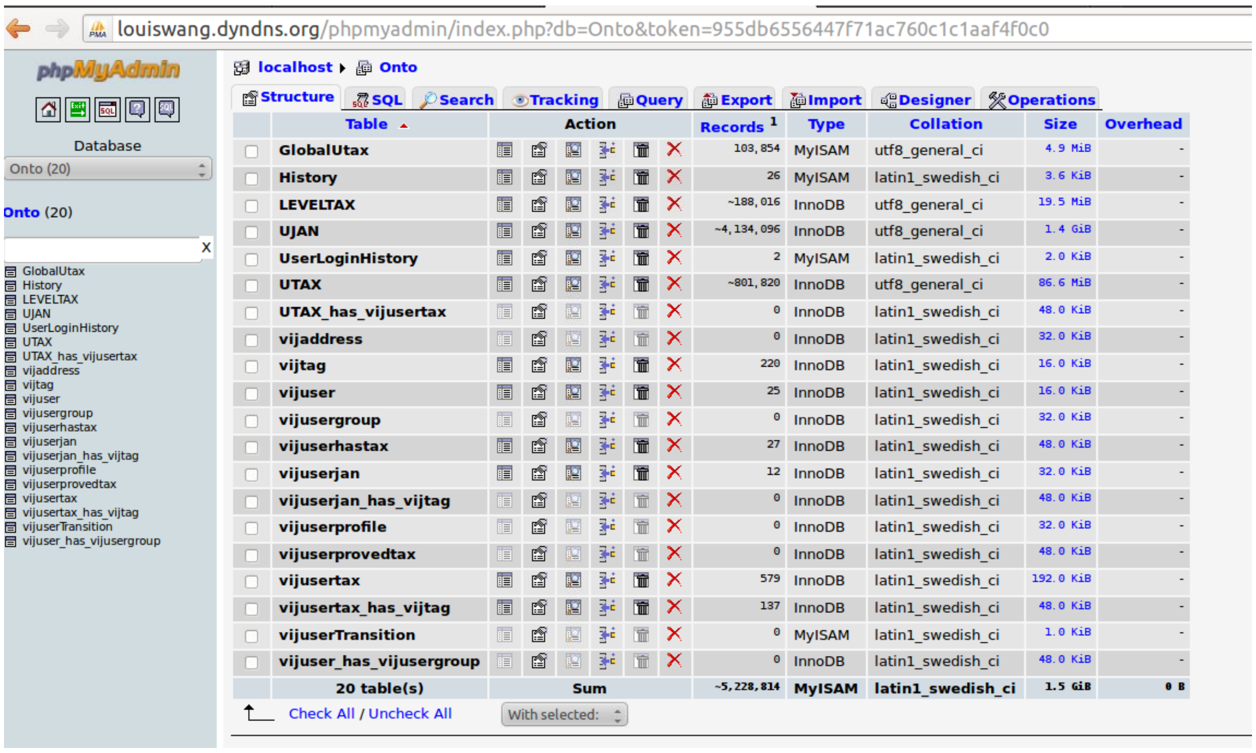


Table	Action	Records [1]	Type	Collation	Size	Overhead
GlobalUtax		103,854	MyISAM	utf8_general_ci	4.9 MiB	-
History		26	MyISAM	latin1_swedish_ci	3.6 KiB	-
LEVELTAX		~188,016	InnoDB	utf8_general_ci	19.5 MiB	-
UJAN		~4,134,096	InnoDB	utf8_general_ci	1.4 GiB	-
UserLoginHistory		2	MyISAM	latin1_swedish_ci	2.0 KiB	-
UTAX		~801,820	InnoDB	utf8_general_ci	86.6 MiB	-
UTAX_has_vijusertax		0	InnoDB	latin1_swedish_ci	48.0 KiB	-
vijaddress		0	InnoDB	latin1_swedish_ci	32.0 KiB	-
vijtag		220	InnoDB	latin1_swedish_ci	16.0 KiB	-
vijuser		25	InnoDB	latin1_swedish_ci	16.0 KiB	-
vijusergroup		0	InnoDB	latin1_swedish_ci	32.0 KiB	-
vijuserhastax		27	InnoDB	latin1_swedish_ci	48.0 KiB	-
vijuserjan		12	InnoDB	latin1_swedish_ci	32.0 KiB	-
vijuserjan_has_vijtag		0	InnoDB	latin1_swedish_ci	48.0 KiB	-
vijuserprofile		0	InnoDB	latin1_swedish_ci	32.0 KiB	-
vijuserprovedtax		0	InnoDB	latin1_swedish_ci	48.0 KiB	-
vijusertax		579	InnoDB	latin1_swedish_ci	192.0 KiB	-
vijusertax_has_vijtag		137	InnoDB	latin1_swedish_ci	48.0 KiB	-
vijuserTransition		0	MyISAM	latin1_swedish_ci	1.0 KiB	-
vijuser_has_vijusergroup		0	InnoDB	latin1_swedish_ci	48.0 KiB	-
20 table(s)	Sum	~5,228,814	MyISAM	latin1_swedish_ci	1.5 GiB	0 B

Figure 7.8: Database tables

and deployment tool. Besides the Spring framework and Hibernate library, we also use the apache common library which offers basic file operations. In addition to that, we apply Apache Lucene to do the FULL TEXT indexing as mentioned in last Chapter.

7.3 Conclusion

We are still actively developing and adding features in our KAM implementation. We have developed several prototype mobile phone applications. The ontology reasoning and personal email systems are the most demanding features to be expected in next stage. The current KAM implementation laid a solid ground for future development.

Chapter 8

Conclusion and Discussion

In this thesis, I explained the way we built a context awareness model in the Knowledge Advantage Machine by utilizing ontology pattern.

The KAM is intended to effectively help people find correct information with less effort. In the KAM model, all resources are unique and identified by URI. Based on this feature, I introduce the knowledge object concept that each general resource in KAM is abstracted into a knowledge object JAN. This provides an abstraction layer upon resources. Even for the same resource, this abstraction layer ensures the uniqueness of a JAN for different users. The JAN object is constructed according to the IEEE LOM standard.

To better organize JAN, I use the ROA architecture as the resource infrastructure for the whole model. The ROA architecture not only allows us to neglect the issues caused by resource duplication but also eliminates the issues caused by resource control. All operations supported in our ROA implementation are stateless. This ensures that our distributed architecture can expand seamlessly across different components. Also the ROA resource naming strategy provides us a more effective way to check resource consistency and organization. We organize the resources based on the Ontology architecture in that JANs are categorized into taxonomies and taxonomies are laid out in a hierarchy structure.

From a user's perspective, facilitated by help from agents, leveraging knowledge becomes more effective. According to users' behavior analysis and an existing knowledge repository, KAM creates user context and distills knowledge along with context changes. In my approaches, I model context according to user ontology patterns. In KAM, the user ontology consists of two parts. One is a web user ontology created through the KAM web interface and the other is built upon a user filesystem through KAM agents. Besides the user ontology, KAM also stores a universal ontology that relies on the data distilled from Open Directory Project. The universal ontology acts as a solid base for user ontology. The user ontology construction process involves three steps: (1) create taxonomy. In this step, a taxonomy is manually created byreferring to the universal ontology, (2) JAN abstraction and mark up process. In this step, a JAN is manually created or cited from the web. Afterward, this JAN is processed by the Discovery Agent and Markup Agent to generate a JAN annotation. (3) "Call in once" discover process. In this step, the search procedures are carried inside the knowledge repositories and the return results reflect the current user context and ontologies. An important procedure in step two is to generate JAN annotations. I developed my own keyphrase extraction algorithm which combines statistical methods and heuristic rules together. It replaced our old implementation that used the KEA algorithm. The new VKE algorithm is more suitable than a traditional tf*idf approach introduced in the KEA algorithm. Comparatively, the VKE algorithm, illustrated in Chapter Five, doesn't require any training process. The random process can be injected in any predefined distribution. It also doesn't rely any language features.

Another emphasis lies in step three that requires that the result reflect the user context. I created two models to detect user context once we have the user ontology in place. First is the timeline model. Based on the distribution of user activities, we can select the taxonomy with the highest probability within a certain period as the user current context. The second relies on monitoring user behaviour. When a transition happens between user taxonomies, KAM calculates an interest scores for all user taxonomies. Based on these scores and their stationary probability, we can predict the user context switch and taxonomy to which a user will move.

The current implementation consists a web interface and standalone application. The web interface is a cloud based solution of KAM. All intelligent agents run as web services which can be accessed anywhere. The standalone application mainly focuses on the personal user file system part. All agents runs as normal processes and the data are shared through the same data layer as web interface.

Till this point, KAM is already ready to serve use. However we are still actively working in make it more mature and functional. There are several expected features that are under development, like ontology reasoning and use KAM to cover personal email system. The current KAM implementation laid solid ground for future development. KAM right now mainly focuses on textural structural data. However, the new trend of internet development with video and audio is becoming more popular [?][?] and comprise over 25% percent internet content. Also, a lot of emerging social commerce shifts the marketing focus to social media as well. All these demand us to add more features into KAM to support more content types. These are all my future research interests.

References

[1] Ahmed Louri and Jongwhoa Na, "Design of an optical content-addressable parallel processor for expert systems," *Appl. Opt.*, vol. 34, no. 23, pp. 5053–5063, Aug 1995.

[2] Efraim Turban, *Decision Support and Expert Systems: Management Support Systems*, Prentice Hall PTR, Upper Saddle River, NJ, USA, 2nd edition, 1990.

[3] Y Cho, "A personalized recommender system based on web usage mining and decision tree induction," *Expert Systems with Applications*, vol. 23, no. 3, pp. 329–342, 2002.

[4] Peter Politakis and Sholom M. Weiss, "Using empirical analysis to refine expert system knowledge bases," *Artif. Intell.*, vol. 22, no. 1, pp. 23–48, 1984.

[5] Thomas R. Gruber, "Toward principles for the design of ontologies used for knowledge sharing," in *IN FORMAL ONTOLOGY IN CONCEPTUAL ANALYSIS AND KNOWLEDGE REPRESENTATION, KLUWER ACADEMIC PUBLISHERS, IN PRESS. SUBSTANTIAL REVISION OF PAPER PRESENTED AT THE INTERNATIONAL WORKSHOP ON FORMAL ONTOLOGY*. 1993, Kluwer Academic Publishers.

[6] Fabrizio Sebastiani, "Machine learning in automated text categorization," *ACM Comput. Surv.*, vol. 34, pp. 1–47, March 2002.

[7] Susan Dumais, John Platt, David Heckerman, and Mehran Sahami, "Inductive learning algorithms and representations for text categorization," in *Proceedings of the seventh international conference on Information and knowledge management*, New York, NY, USA, 1998, CIKM '98, pp. 148–155, ACM.

[8] Andrew McCallum and Kamal Nigam, "A comparison of event models for naive bayes text classification," in *IN AAAI-98 WORKSHOP ON LEARNING FOR TEXT CATEGORIZATION*. 1998, pp. 41–48, AAAI Press.

[9] Sholom M. Weiss, Fred J. Damerau, David E. Johnson, Frank J. Oles, and Thilo Goetz, "Maximizing text-mining performance," *IEEE Intelligent Systems*, vol. 14, pp. 63–69, 1999.

[10] Kamal Nigam, Andrew Kachites Mccallum, Sebastian Thrun, and Tom Mitchell, "Text classification from labeled and unlabeled documents using em," in *Machine Learning*, 1999, pp. 103–134.

[11] Inderjit S. Dhillon, Subramanyam Mallela, and Rahul Kumar, “A divisive information-theoretic feature clustering algorithm for text classification,” *Journal of Machine Learning Research*, vol. 3, pp. 1265–1287, 2003.

[12] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “Wireless sensor networks: a survey,” *Computer Networks*, vol. 38, pp. 393–422, 2002.

[13] Hyoung R. Kim and Philip 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, 2003, pp. 101–108, ACM.

[14] Edward Feigenbaum and Pamela McCorduck, *The fifth generation: artificial intelligence and Japan’s computer challenge to the world*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1983.

[15] Howard Jay Siegel, *Interconnection networks for large-scale parallel processing: theory and case studies (2nd ed.)*, McGraw-Hill, Inc., New York, NY, USA, 1990.

[16] H. J. Levesque and R. J. Brachman, *A fundamental tradeoff in knowledge representation and reasoning*, Readings in Knowledge Representation. Morgan Kaufmann, 1985.

[17] William J. Clancey, “The knowledge level reinterpreted: Modeling how systems interact,” *Machine Learning*, vol. 4, pp. 285–291, 1989.

[18] DA Waterman, “A Guide to Expert Systems,” 1986.

[19] R. R. Studer, R. Benjamins, and D. Fensel, “Knowledge engineering: principles and methods,” *Data and knowledge engineering*, vol. 25, pp. 161–197, 1998.

[20] Thomas R. Gruber, “A translation approach to portable ontology specifications,” *KNOWLEDGE ACQUISITION*, vol. 5, pp. 199–220, 1993.

[21] *Ontology theory, management and design : advanced tools and models*, Information Science Reference, Hershey, PA, 2010.

[22] 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.

[23] Deborah L. McGuinness and Paulo Pinheiro da Silva, “Explaining answers from the semantic web: the inference web approach,” *Web Semant.*, vol. 1, no. 4, pp. 397–413, 2004.

[24] Ian Horrocks and Ulrike Sattler, “Ontology reasoning in the shoq(d) description logic,” in *In Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001*. 2001, pp. 199–204, Morgan Kaufmann.

[25] Natalya F. Noy and Michel Klein, “Ontology evolution: Not the same as schema evolution,” *Knowledge and Information Systems*, vol. 6, pp. 428–440, 2003.

[26] Leo Sauermann, Gunnar Aastr, Malte Kiesel, Heiko Maus, Dominik Heim, Danish Nadeem, Benjamin Horak, and Andreas Dengel, "A.: Semantic desktop 2.0: The gnowsis experience," in *International Semantic Web Conference. Volume 4273 of Lecture Notes in Computer Science*. 2006, pp. 887–900, Springer.

[27] Wolfgang Woerndl and Georg Groh, "A social item filtering approach for a mobile semantic desktop application," 2008.

[28] Henry Lieberman, "Letizia: An agent that assists web browsing," in *INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE*, 1995, pp. 924–929.

[29] Christoph G. Thomas and Gerhard Fischer, "Using agents to personalize the web," in *IUI '97: Proceedings of the 2nd international conference on Intelligent user interfaces*, New York, NY, USA, 1997, pp. 53–60, ACM.

[30] Michael Pazzani, Jack Muramatsu, and Daniel Billsus, "Syskill & webert: Identifying interesting web sites," in *In Proc. 13th Natl. Conf. on Artificial Intelligence*, 1998, pp. 54–61.

[31] Francisco Tanudjaja and Lik Mui, "Persona: A contextualized and personalized web search," in *In Proc. of the 35th Annual Hawaii International Conference on System Sciences*, 2001, p. 67.

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

[33] Stuart E. Middleton, Nigel R. Shadbolt, and David C. De Roure, "Capturing interest through inference and visualization: ontological user profiling in recommender systems," in *K-CAP '03: Proceedings of the 2nd international conference on Knowledge capture*, New York, NY, USA, 2003, pp. 62–69, ACM.

[34] Chien Chin Chen, Meng Chang Chen, and Yeali Sun, "Pva: A self-adaptive personal view agent," 2002.

[35] Joana Trajkova and Susan Gauch, "Improving ontology-based user profiles," 2004.

[36] Open Directory Project, "Open directory project website,http://dmoz.org," April 2002.

[37] Liren Chen and Katia 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, 1998, pp. 132–139, ACM.

[38] Stuart E. Middleton, Nigel R. Shadbolt, and David C. De Roure, "Ontological user profiling in recommender systems," *ACM Trans. Inf. Syst.*, vol. 22, pp. 54–88, January 2004.

[39] R. Reddy, L. Wang, S. Reddy, S. Devalapalli, G. Sasanka, S. Macha, S. Teja, R. Doppalapudi, and J. Yu, "Vijjana: A pragmatic model for collaborative, self-organizing, domain centric knowledge networks," in *IKE*, 2008, pp. 116–121.

[40] Learning Technology Standards Committee of the IEEE, “Draft standard for learning technology - learning object metadata,” Tech. Rep., IEEE Standards Department, New York, July 2002.

[41] Leonard Richardson and Sam Ruby, *Restful Web Services*, O’Reilly Media, 2007.

[42] “Semantic web 3.0,http://www.w3.org/2001/sw/wiki/main_page,” .

[43] S. Devalapalli, R. Reddy, L. Wang, and S. Reddy, “Markup and validation agents in vijjana - a pragmatic model for collaborative, self-organizing, domain centric knowledge networks,” in *WEBIST*, 2009, pp. 263–269.

[44] A. McCallum and K. Nigam, “A comparison of event models for naive bayes text classification,” 1998.

[45] Yufei Yuan and Michael J. Shaw, “Induction of fuzzy decision trees,” *Fuzzy Sets and Systems*, vol. 69, no. 2, pp. 125 – 139, 1995.

[46] Makoto Nagao, “A framework of a mechanical translation between japanese and english by analogy principle,” in *Proc. of the international NATO symposium on Artificial and human intelligence*, New York, NY, USA, 1984, pp. 173–180, Elsevier North-Holland, Inc.

[47] Yi fang Brook Wu, Quanzhi Li, Razvan Stefan Bot, and Xin Chen, “Domain-specific keyphrase extraction,” in *CIKM ’05: Proceedings of the 14th ACM international conference on Information and knowledge management*, New York, NY, USA, 2005, pp. 283–284, ACM.

[48] Peter D Turney, “Learning algorithms for keyphrase extraction,” *Information Retrieval*, vol. 2, pp. 303–336, 2000.

[49] Eibe Frank, Gordon W. Paynter, Ian H. Witten, Carl Gutwin, and Craig G. Nevill-manning, “Domain-specific keyphrase extraction,” 1999, pp. 668–673, Morgan Kaufmann Publishers.

[50] Julie B. Lovins, “Development of a stemming algorithm,” *Mechanical Translation and Computational Linguistics*, vol. 11, pp. 22–31, 1968.

[51] Martin F. Porter, “An algorithm for suffix stripping,” *Program*, vol. 14, no. 3, pp. 130–137, 1980.

[52] B. Krulwich and C. Burkley, “Learning User Information Interests Through Extraction of Semantically Significant Phrases,” in *Proceedings of the AAAI Spring Symposium on Machine Learning in Information Access*. 1996, AAAI Press, Stanford, CA.

[53] Adam L. Berger, Vincent J. Della Pietra, and Stephen A. Della Pietra, “A maximum entropy approach to natural language processing,” *Comput. Linguist.*, vol. 22, no. 1, pp. 39–71, 1996.

[54] Alberto MuÃśoz, "Compound key word generation from document databases using a hierarchical clustering art model.," *Intell. Data Anal.*, vol. 1, no. 1-4, pp. 25–48, 1997.

[55] Andrew McCallum, Dayne Freitag, and Fernando Pereira, "Maximum entropy Markov models for information extraction and segmentation," in *Proc. 17th International Conf. on Machine Learning.* 2000, pp. 591–598, Morgan Kaufmann, San Francisco, CA.

[56] Gerard Salton and Christopher Buckley, "Term-weighting approaches in automatic text retrieval," in *INFORMATION PROCESSING AND MANAGEMENT*, 1988, pp. 513–523.

[57] Siddhartha Chib and Edward Greenberg, "Understanding the metropolis-hastings algorithm," *The American Statistician*, vol. 49, no. 4, pp. 327–335, Nov. 1995.

[58] "Monte carlo introduction wiki, http://en.wikipedia.org/wiki/monte_carlo_method," .

[59] Christophe Andrieu, Nando de Freitas, Arnaud Doucet, and Michael I. Jordan, "An introduction to mcmc for machine learning.," *Machine Learning*, vol. 50, no. 1-2, pp. 5–43, 2003.

[60] "Variance analysis, http://en.wikipedia.org/wiki/analysis_of_variance," .

[61] Hele mai Haav and Tanel lauri Lubi, "A survey of concept-based information retrieval tools on the web," in *In 5th East-European Conference, ADBIS 2001*, 2001, pp. 29–41.

[62] "Apache lucence, http://lucene.apache.org/core/," .

[63] Verayuth Lertnattee and Thanaruk Theeramunkong, "Effect of term distributions on centroid-based text categorization," *Inf. Sci. Inf. Comput. Sci.*, vol. 158, pp. 89–115, January 2004.

[64] Amit Singhal, Chris Buckley, and Mandar Mitra, "Pivoted document length normalization," in *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, 1996, SIGIR '96, pp. 21–29, ACM.

[65] Jeffrey Heer, Stuart K. Card, and James A. Landay, "prefuse: a toolkit for interactive information visualization," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, New York, NY, USA, 2005, CHI '05, pp. 421–430, ACM.

[66] Rod Johnson, Juergen Hoeller, Alef Arendsen, Thomas Risberg, and Dmitriy Kopylenko, *Professional Java Development with the Spring Framework*, Wrox Press Ltd., Birmingham, UK, UK, 2005.

[67] Christian Bauer and Gavin King, *Java Persistence with Hibernate*, Manning Publications Co., Greenwich, CT, USA, 2006.

[68] "Content marketing state,http://blog.outbrain.com/2012/03/state-of-content-marketing-2012.html," .

[69] “Content type survey, http://www.marketingcharts.com/interactive/college-students-online-video-use-dramatically-surpasses-general-populations-1599/survey-u-online-video-content-type-watchedgif/,” .