An Information-Theoretic Approach to Interest Matching

by

Waikit Koh

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2001

Author ....
Department of Electrical Engineering and Computer Science
May 23, 2001

Certified by ....
Peter Szolovits
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by ....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

An Information-Theoretic Approach to Interest Matching

by

Waikit Koh

Submitted to the Department of Electrical Engineering and Computer Science
on May 23, 2001, in partial fulfillment of the
requirements for the degree of
Master of Engineering

Abstract

The Internet has brought a new meaning to the term *communities*. Geography is no longer a barrier to international communications. However, the paradigm of meeting new interesting people remains entrenched in traditional means; meeting new interesting people on the Internet still relies on chance and contacts. This thesis explores a new approach towards matching users in online communities in an effective fashion.

Instead of using the conventional feature-vector scheme to profile users, each user is represented by a personalized concept hierarchy (or an ontology) that is learnt from the user's behavior in the system. Each concept hierarchy is then interpreted within the Information Theory framework as a probabilistic decision tree. The matching algorithm uses the Kullback-Leiber distance as a measure of deviation between two probabilistic decision trees. Thus, in an online community, where a personalized concept hierarchy represents each user, the Kullback-Leiber distance imposes a full-order rank on the level of similarity of all the users with respect to a particular user in question.

The validity and utility of the proposed scheme of matching users is then applied in a set of simulations, using the feature-vector-overlap measure as a baseline. The results of the simulations show that the Kullback-Leiber distance, when used in conjunction with the concept hierarchy, is more robust to noise and is able to make a stronger and more distinctive classification of users into similar groups in comparison to the conventional keyword-overlap scheme. A graphical agent system that relies upon the ontology-based interest matching algorithm, called the Collaborative Sanctioning Network, is also described in this thesis.

Thesis Supervisor: Peter Szolovits
Title: Professor of Electrical Engineering and Computer Science

Acknowledgments

If I have seen further than I previously could have, it is because I have stood upon the shoulders of giants.

I have taken the liberty to modify Sir Isaac Newton's famous quote to express my gratitude to all who taken time out to support this project. In particular, I would like to thank Professor Szolovits for taking me into the research group and under his guidance. I am indebted to Lik for his invaluable teachings, guidance and many nights of discussions. I would also like to thank Dr. Mohtashemi for her invaluable feedback on the theoretical aspects of this thesis.

This thesis would not have been possible without the inspirations from people I have had the pleasure to know.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A community is composed of a group of people with varying degrees of expertise in which experts attempt to help the novices improve their skills [8]. The structure of a community usually depends on the specificity of the interests of the people that comprises the community. However, in general, a community can be decomposed taxonomically into areas of interests pursued by members of the community. Within each area of interests, members can be classified by the level of expertise or depth of knowledge of that particular area. The divisions between areas of interest are usually not well-defined; areas of interest are related to each other in various ways. As a result, opinions of an expert (in his or her area of interest) usually matter more to members whose areas of interest are closely related to that of the expert than to members whose area of interests are very much different.

In offline communities, meeting new members in the community occurs either by chance or through other contacts the member presently knows. Since the objective of a member in the community is to locate other members who share similar interest to interact with, the member must potentially interact with all the members to be able to deciding on the desired subset of members. Moreover, in the process of finding suitable members to interact with, meeting new members is only the first step; a long process of interaction and communication usually ensues in determining compatibility between members. As such, finding similar users in an offline community is a time consuming process.

The following scenario illustrates the problems of offline communities,

> There has been a disease outbreak in South East Asia. You have been given the responsibility of assembling a team of suitable medical personnel, engineers and environmentalists to contain the disease outbreak. The disease is epidemic in nature and every day, hundreds of people die from it.
>
> You do not know who is suitable for the task. Criteria for suitability are complicated, ranging from familarity with South East Asian climate, food and commonly associated diseases, to experience in working with equipment that the task requires. As such, you approach present individuals you may know and ask them for their opinions on who they might know that may be suitable for the task. You proceed to contact these new individuals and the process goes on until you have found a sufficiently large group of individuals to interview and filter to narrow down to a suitable team.
>
> The process of discovering and interviewing new people is time consuming and expensive. After a week, you have assembled a team that you think is suitable for the task. Meanwhile, more people die by the day and your team has yet to leave for the region.

If there had been a more efficient way of locating suitable individuals in the community dynamically, the time to assembling a team can be reduced and more lives could have been saved.

The introduction of the Internet in 1994 brought about a change in the traditional conception of a community. Physical distance is no longer an important constraint in determining membership of a community. It has also allowed its users to communicate with one another with minimal cost in time and money. Information flow can take place easily through synchronous means, such as instant messaging, chatrooms and telephony, or through asynchronous means, such as email and articles published on the Web and newsgroups. With the recent poliferation of peer-to-peer networks, such as Napster[1] and Freenet[2], the Internet has also envolved into a resource-sharing medium between users.

Online communities also have an important advantage over traditional offline communities; the behavior of users can be logged and analyzed to provide additional information about users that are previously unavailable prior to the Internet. A profile of each user can be built up from his or her online behavior, such as items purchased over the Internet and access patterns, to determine new content the user might be interested in. As such, instead of relying on the user to discover new interesting content, the system can push new information to the user and speed up the match making process. This is exemplifed by targeted advertisements, personalized content and shopping recommendations.

Despite the improvements in matching content to users, there has been relatively little work on matching users to users. Nor is there an equivalent of a search engine in the space of user-to-user matching. As of the year 2000, there are an estimated 400 million users worldwide. The Internet thus potentially provides each user with a virtual neighborhood of 400 million neighbors. Although this allows for communication and collaboration between users of an unprecedented magnitude, there is still a problem of finding suitable individuals to communicate with. Analogous to the problem of information overload, where users have to wade through a sea of content to locate the relevant information sources, in the context of online communities, users have to go through a huge list of other users to find the subset of similar or relevant neighbors. This problem is further compounded by the fact that in order to determine if two individuals are compatible, they must interact over an extended period of time.

Just as information retrieval solutions (such as search engines) are essential to information systems, there is an obvious need for an analogous software component in online communities that help a user to locate other compatible users who share similar interests. By off-loading the laborious task of locating similar users into the system, the system can search through its database of users and automate user

[1] http://www.napster

[2] http://freenet.sourceforge.net

discovery and match-making. In the disease outbreak scenario described previously, the automated user profiling and search would have hastened the process of assemling a team and thus saved many lives.

In addition to automated match-making services, the ability to quantify similarity between users allows other important services to be built upon it. An important example is Collaborative Sanctioning [12], currently being developed at MIT's Laboratory for Computer Science. Collaborative Sanctioning is a new approach towards reputation management in online communities. Each user's reputation in a community is personalized with respect to other users. However, reputation is also a contextualized quantity. For example, a MIT professor of AI clearly commands a high reputation in the community of AI researchers. However, within the community of food connoisseurs, his or her opinions may not be valued highly. The ability to measure the reliability of the professor's opinions becomes harder in fields that may be related to AI in varying degree such as computer science, robotics, networking. How should the professor's opinions be evaluated in each of these fields? These questions are examples of two important questions in reputation management:

- Context - how does reputation of an individual change from one area of interest to another?
- User - how does reputation of an individual change from the viewpoint of one individual to another?

Clearly, the first step to solving the two problems is the ability to quantify *similarity* between *contexts* and between *users.* The process of matching resources is called *interest matching.*

1.1 Previous Work

Works on interest-matching can be classified into two categories:

- Content-to-User
- User-to-User

1.1.1 Content-to-User Matching

Content-to-User matching focuses on matching self-descriptive documents to users. Examples of such match-making services are personalized content delivery, search engines and personalized search engines. The following paragraphs describe examples of each of these categories of services.

PointCast

PointCast [13] is a personalized content delivery service, in which contents that are classified under categories the user has indicated interests in are pushed to the user's desktop. The classification of contents and the process of learning the user's interest are performed manually. The taxonomy of the content classification is a simple singly-connected tree with minimal breadth and depth.

The process of user profiling and interest matching is relatively unsophisticated: a match occurs when content that are classified under categories that the user has indicated interests in. As such, instead of a true personalized service, PointCast functions more of a first-pass information filter.

Altavista

Altavista[3] is one of the more popular search engines for locating websites that the user may be interested in. Keywords from the user are marshalled into a feature vector that represents the user's interest in the search query. The search engine contains a list of index websites where each website is also represented by a feature vector, composed from important words used in the website.

The process of matching compares the feature vector of the search query against the feature vectors of the document. The degree of alignment betwen the two feature vectors indicate the degree of relevance of the document to the search query of the user. The search engine then returns the links to the websites that the matching algorithm has indicated to be highly relevant to the user's

[3]http://www.altavista.com

query. This approach of matching content-to-user requires minimal input from the user; therefore, it has the advantage of simplicity and speed. However, in representing each website and the query with feature vectors, search engines ignores conceptual relations between keywords. As a result, search engines fail to return links to websites that may be highly relevant to a query while written up in a vocabulary of words that is different from the user's input.

Ontology Based Personalized Search

Because each keyword has a different mearning, depending on the context, search engines often return results that may be completely irrelevant to the user's needs. The Ontology Based Personalized Search [14] improves upon commerical search engines by including the user's profile in the matching process. The user's profile can be learned through explicit feedback about the quality of websites or observing the user's access patterns when surfing the web. The user's profile is also another feature vector, with each keyword representing the context that has been associated with the user's past search queries. The feature vector is combined with the search query into a feature vector that represents the user's needs more accurately.

Although Personalized Search introduces the use of user profiles, the matching algortihm remains fundamentally the same as that used in most search engines. As such, it also suffers from the inability to capture semantic relations between keywords.

1.1.2 User-to-User Matching

User-to-user matching focuses on matching up users in a community. By matching up users, other value-added services can be built upon groups of similar users such as automated recommendations. Some content, such as music, movies or some document formats, are difficult to analyze and do not allow automatic extraction of representative keywords. As such, the conventional method that relies on content description to matching content to user cannot be applied. Instead, content-to-user matching can be built on top of user-to-user matching by assuming that users that are similar to one another tend to be interested in the same content. The following examples illustrate present works on matching users to users.

Amazon.com

Amazon.com's Recommendation Service [1] recommends new items to its customers by analyzing the customer's purchase history and comparing it to other customers. It utilizes explicit rating feedback on purchased items from its customers so as to finetune the recommendations. The rationale behind the service is that if two customers have the same preferences in the same area of interests, it is quite likely that a customer will be interested in the item that the other customer has rated highly.

GroupLens

GroupLens [16] was a pioneering work in collaborative filtering and recommendation services.It was conceived as a system that can help netnews readers find articles and increase the chances that they would find useful reading. Although the purpose of GroupLens is to *help people make choices based on opinions of other people*, the underlying theme was to be able to predict the interest of a reader in an article based on the heurisitc that *people who agreed (on the quality of a common news item they have read) will probably agree again.*

Amazon.com and GroupLens are two examples of collaborative systems [4] in which instances are rated and used as a heuristic in predicting future interests of the user. Although these systems use the notion of similarity implicitly, they are not designed for matching users to users explicitly. As a result, higher level services (such as reputation management) cannot be built on top of these systems.

Yenta

Yenta [7] is a system, developed at the MIT Media Lab, that allows automated generation of clusters of users who are interested in the same topics. Each

[4]See [2] for more references

user is represented by a software agent in the virtual community. First contact between users is modeled by the distributed multi-agent nature of the system. Yenta is designed to solve the problem of meeting new people in the community and to improve collaboration between groups of people who are working on similar areas of interests.

Retsina

Unlike Yenta, Retsina [17] is a general match-making system for software agents. Agents represent resources in the network; they may represent human users in the community or resources such as servers and printers. Agents communicate with each other in a language called LARKS. The semantics of each message is closely related to agent communication languages such as Knowledge Query and Manipulation Language (KQML) [6] and Agent Commmunication Language developed by the Foundation of Intelligent Physical Agents (FIPA-ACL) [3].

When the user of a system desires to locate a resource, he creates a logical statement that contains keywords that may describe the resource and the system will automatically match the statement with the resource description it current knows of. The system will then return the directions to resources that can fulfil the needs of the user.

Yenta and Retsina are two important projects in the research of matching users to users. However, their approaches to matching are also based on the feature vector technique employed in content-to-user matching services. The *cosine similarity* equation at the core of these interest matching approaches is

$$Similarity = \frac{\overrightarrow{F_1} \cdot \overrightarrow{F_2}}{|\overrightarrow{F_1}||\overrightarrow{F_2}|} \tag{1.1}$$

This equation has remained relatively unchanged since its introduction in *Information Retrieval* [18] in the 1975. Although this equation is the engine behind many information search solutions, the validity of its application has often been questioned.

For the equation to be semantically sound, each feature in the feature vector has to be conceptually disjoint to other features in the vectors. However, most applications clearly forsake this constraint for simplicity of use.

The cosine similarity algorithm has the advantage of simplicity. However, in matching users to users, the design concerns are different. There are no semantic tags attached to each user. Moreover, representing a user with a simple set of keywords is insufficient; it lacks the proper semantics to describe the complex nature of a user. Feature vectors ignore conceptual relations between keywords. For example, *Porshe* and *Mercedes* are clearly strongly related to each other in many ways. However, in the feature vector representation, they are assumed to be disjoint concepts. This is usually not a problem in the space of content-to-user matching, because the document contains many words. Therefore, it is likely that it contains many keywords that are related. However, in user-to-user matching, using feature vectors to represent each user and the cosine similarity as the matching algorithm is clearly lacking and semantically unsound; it will not be able to match users who have interests in closely related fields but are described by different keywords. The cosine similarity is also highly restrictive and does not scale to other representations of the user, such as neural networks and semantic nets.

1.2 Thesis Contributions

The objective of this thesis is to propose the use of a personalized concept hierarchy - or an ontology - to represent each user and a novel interest-matching algorithm that utilizes the Kullback-Leibler distance and other concepts in Information Theory to match users in online communities. User representation and profiling is properly defined and applied within the framework of ontologies to yield a more adequate form of user profiles. The Kullback-Leiber distance is then shown to be a general approach to matching user profiles and is independent of the choice of user representation. The study includes a set of simulation data to determine the effectiveness of the proposed algorithm compared to the traditional means of matching users.

Chapter 2

Theory

In order to arrive at a meaningful interest matching algorithm, it is important to review some social aspects of online communities and lay down guidelines for performing interest matching.

2.1 Considerations

In many communities (be it online or offline), the concept of similarity between users is not well-defined. To be able to match users, the detailed psychological profile and the history of social activities of each user must be known completely. Although, it is difficult to obtain the former pre-requisite, the Internet has made it easier to track activities of the user. The problem of profiling users is further compounded by the fact that every user's interests change over time and are dynamic. As such, to match users, a third party observer has to track the activities of each user and use the data collected to estimate the user's interests.

It is in everyone's best interest to cooperate and share resources. This allows for greater progress to be made compared to that by a singular effort. As such, users usually interact with others who engage in similar activities to obtain and/or to provide assistance with obstacles encountered. This provides a useful social observation that *users who are alike tend to engage in similar activities.* The converse of the statement is not necessarily true; it is not clear that *users who engage in similar activities are* *alike.* However, many companies, such as Amazon, accepted the previous statement as a social heuristic and have used it to great success. The heuristic thus allows the observer to cluster users into groups. Each group consists of users who tend to engage in common activities, and therefore may be assumed to share similar interests.

In a computerized environment, where each user is represented by an *interest profile* and the match making service is an algorithm, the heuristic must be refined into a more precise mathematical statement. In this thesis, an *interest profile* is a description of the entity's interests. It can be constructed either from explicit feedback from the user, or implicit observations of the activities of the user (See Chapter 3). The interest-matching algorithm then uses the interest profiles to generate a metric that measures the similarity of the interest profiles.

The term *similarity* can be defined so that *User A is similar to User B if and only if User A's interests profile is able to describe User B accurately.* The caveat then lies in the precise quantification of the term *accurately.* The *similarity measure* (or just *similarity*), should satisfy the following guidelines:

- The algorithm must be able to rank every user in the community for the level of similarity with respect to any user.

- For two interests profiles that are exactly the same, the algorithm should generate the value that represents the maximum level of similarity.

Based on the discussion to this point, an error function is an ideal candidate for the interest matching algorithm. However, before selecting an error function, it is necessary to select an appropriate mathematical representation of an interest profile.

2.2 Ontologies as Interest Profiles

Ontologies have garnered much interest in Artificial Intelligence. According to Tom Gruber, an ontology can be defined as "a specification of a conceptualization" [9]. Gruber then continues to explain that

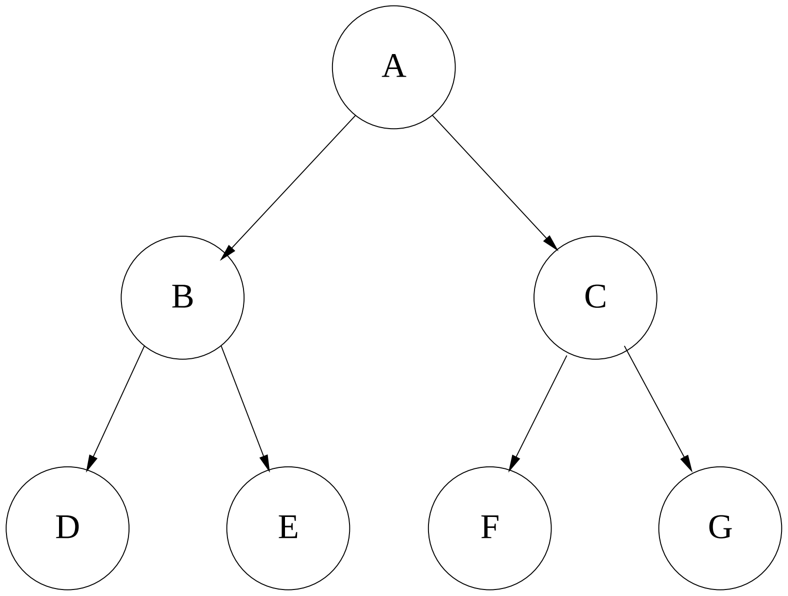


Figure 2-1: A tree representation of a simple ontology

> ... an ontology is a description (like a formal specification of a program) of the concepts and relationships that can exist for an agent or a community of agents. This definition is consistent with the usage of ontology as set-of-concept-definitions, but more general. And it is certainly a different sense of the word than its use in philosophy.

What then qualifies as an ontology? Ontologies can be realized in a variety of ways, ranging from a set of statements about what exists to a graphical representation. In this thesis, an ontology is simplified into a concept hierarchy, an example of which is shown in Figure 2-1.

In this thesis, the semantics of the tree representation of an ontology is defined as follows:

- Each node represents a distinct concept
- Each directed edge represents a *sub-concept* relationship between the parent (the outgoing node) and the child node (the ingoing node)

In Figure 2-1, node A has two child nodes B and C. This means that there is a concept labelled A and it can be classified into 2 sub-concepts, B and C. For simplicity, a tree representing an ontology is assumed to be a singly-connected tree unless stated otherwise. This entails that for each parent node, its child nodes are mutually exclusive. The child nodes of each parent are also assumed to be exhaustive, the child nodes form the complete set of sub-concepts of the parent node.

In the tree representation of the ontology, the root node can be regarded as the simplest classificiation of the user's interests - simply labelled *Interests*. The nodes in the next level of the tree then represent the first broad categorization of the user's interests. Subsequent levels of the tree describe increasingly finer categorization of the interests, right down to the leaf nodes of the tree.

Although the definition of an ontology does not limit the representation of the ontology, a set of keywords, as used in many search engines and other matching algorithms, does not qualify as an ontology. The flat, strutureless nature of a keyword vector (or feature vector) fails to capture the relations between concepts, which are represented by keywords in the vector. Therefore, a simple keyword vector is an inadequate representation of a user.

An ontology can thus be used to specify the user's interests, with the tree-representation is the realization of that specification. However, to construct an ontology for each user is a difficult task. Each user may have a different ontological view of the world; concepts and relations may differ on an individual basis. However, instead of generating a new ontology for each user, each user can adopt a standard, predefined ontology in the community sanctioned by the majority of users. Ontologies like the Yahoo Taxonomy[1], Cyc Knowledge Database[2],and Open Directory Project[3] are built by a large population of users. Therefore these ontologies are universal within each of their own domains. Incidentally, most of the ontologies used in industry are already in a tree representation.

[1]http://www.yahoo.com
[2]http://www.cyc.com
[3]http://www.dmoz.com

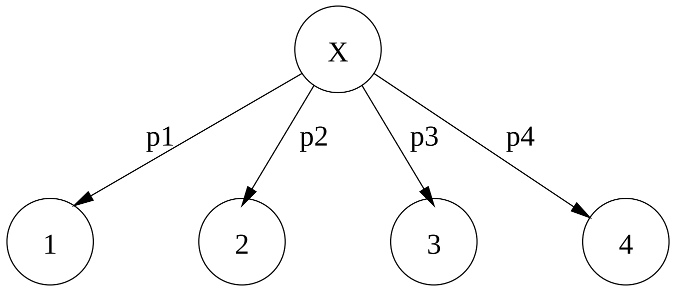
Since each user is to adopt the standard ontology in the community, each ontology must then be personalized to reflect each user's interests and preferences. Since a user's interests may encompass several areas, each pursued with differing degrees of enthusiasm, the personalized ontology must be able to capture such level of detail.

2.2.1 An Ontology as a Probability Tree

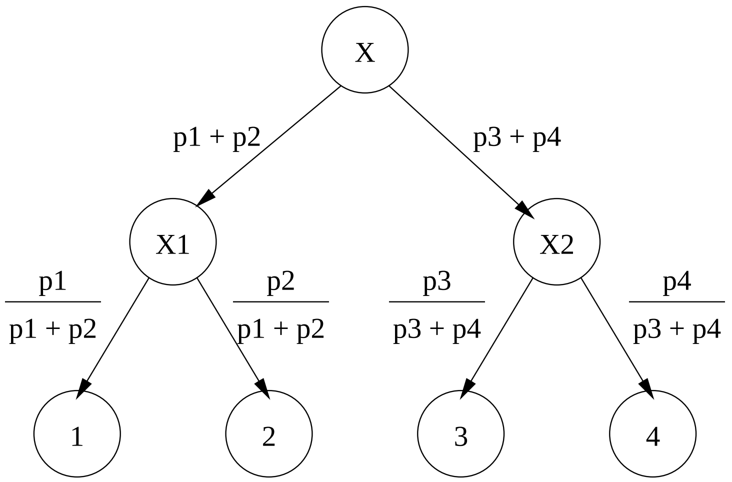
A probability tree is a useful mathematical construct that imposes a hierachical structure on the outcomes of a random variable. Outcomes (usually related) may be grouped in clusters and clusters can be recursively included into larger clusters, thus forming a tree structure. The original random variable is represented by the root node of the tree. Child nodes of the parent node can be considered as outcomes of a random variable represented by the parent node. If the child node itself is the root node of a subtree, it is also a random variable to which the outcomes are its child nodes. Figure 2-2(a) shows a graphical representation of a random variable in terms of a shallow tree. Figure 2-2(b) is a reorganization of the same random variable into clustered outcomes.

As a result of the reorganization, the probability associated with each edge is changed. In Figure 2-2(a), to represent that each outcome has an associated probabiltity that the outcome will be realized, each edge is associated with a probability. Figure 2-2(b), due to the tree structure, each edge is associated with a *conditional probability* in which the probabilty of an outcome (a child node) being realized is dependent on its associated random variable (its parent node).

Along with the previous treatment of the tree representation of the ontology, the probability tree paradigm offers a way to *personalize* the standard ontology. The ontology is a recursive classification of the leaf nodes. The root node represents the *Interests* random variable. Performing an experiment to obtain the outcome of the random variable is equivalent to asking the user, "What is the user's interest?". The outcomes of the random variable are the leaf nodes of the tree. Each parent-child node relationship in the ontology can be quantified by assigning probabilities to each parent-child edge. The probability associated with each outcome is the product of all



(a) A graphical representation of a random variable (root node) and outcomes (child nodes)



(b) A hierachical rearrangement of outcomes

Figure 2-2: A Probability Tree

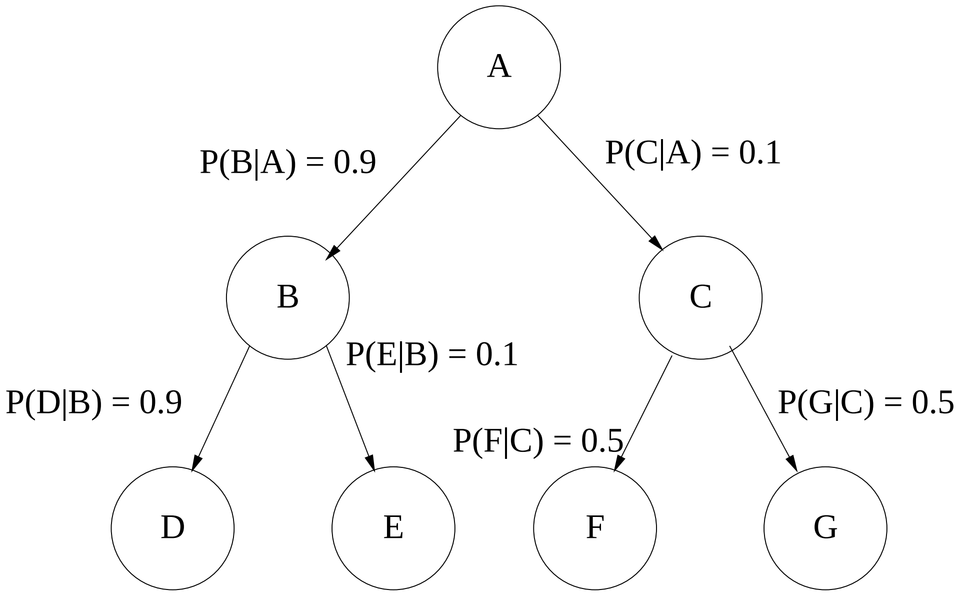


Figure 2-3: An Interest Profile

the probabilities along a single path leading from the root node to the leaf node.

Figure 2-3 shows an interest profile of a user, which is a superposition of a probability tree with Figure 2-1. The ontology in Figure 2-1 is assumed to be the standard ontology of the community to which the user belongs. The semantics of Figure 2-3 is as follows, "Given A, the probability of the user being interested in B is 0.9 and in C is 0.1. Given B, the probability of the user is interested in D is 0.9 and in E os 0.1. Given C, it is equally likely that the user is interested in F and G." From these basic equations depicted in Figure 2-1, other probabilities can be generated by the following rules:

- The probability from node x to another node y further in the tree is the product of all the probability edges that lead from node x to node y. For example, $P_{A,C} = P_{A,B} * P_{B,C} = 0.09$.
- If there are no edge(s) from node x to node y, $P_{x,y} = 0$. For example, given the topic B, there is no way that the user is interested in F, ie. $P_{B,F} = 0$.

An interest profile is thus a singly-connected concept hierarchy with a set of corresponding probabilities associated with each edge in the tree. As such, each user can thus be said to be represented by a probability distribution that spans the same domains of interest as other users do. Having defined the interest profile, the *error function* can be defined as a function that computes the deviations between probabiliy distributions.

2.3 Basics of Information Theory

Information Theory is the study of ensemble properties of probabilistic distributions [5]. In particular, it is used to obtain measures of theoretical efficiency in extracting information out of probabilitistic systems and to answer fundamental questions like “What is the uncertainty of a random variable?”. In the previous section, since each user is represented by a probabilistic distribution over a standard ontology, information theory can be used to derive ensemble properties of each user.

2.3.1 Entropy

Entropy, $H(p)$, where p is the probability function of a random variable X, is a measure of the average amount of information required to describe outcomes of X. It is given by

$$H(p) = -\sum_{i=1}^{N} p(X = x_i) log(p(X = x_i) \tag{2.1}$$

where x_i is one of the N possible outcomes of X. From Equation 2.1, $H(p)$ has a minimum value of 0 when to every experiment, the outcome is always the same. $H(p)$ achieves its maximum value of N when there is an equal likelihood to each of the N possible outcomes occurring. Since $H(p)$ is a measure of uncertainty, a value of 0 represents complete certainty and a value of N represents the maximum uncertainty.

In some situations such as probability trees, there may be benefits in organizing the outcomes of the random variables into groups. The entropy formula can then be

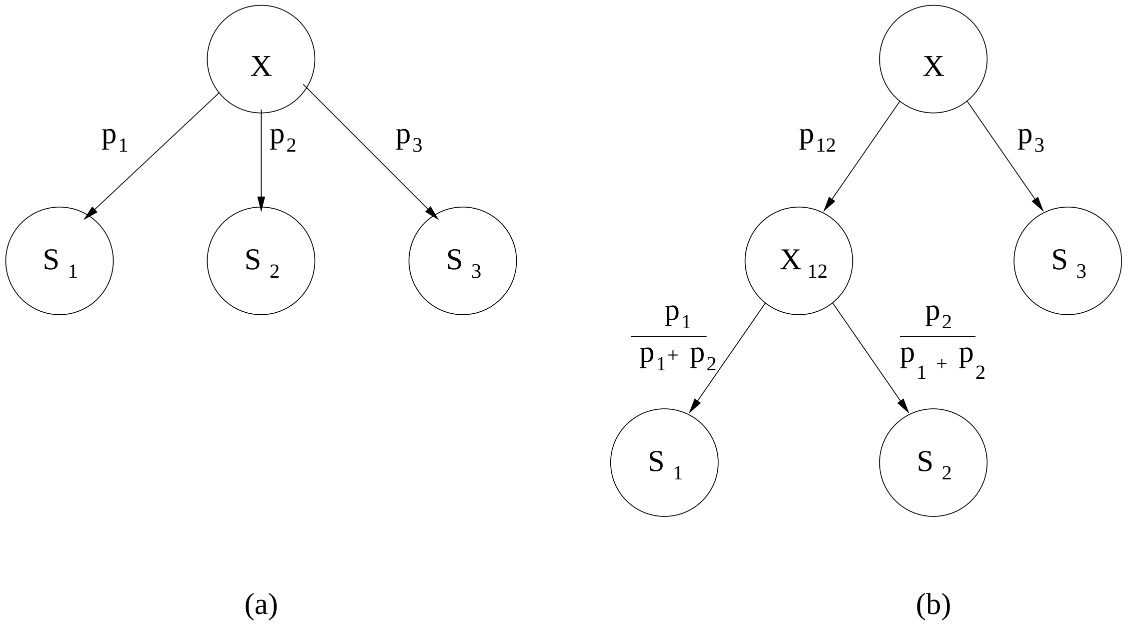


Figure 2-4: (a) Simple Probability Tree (b) Modified Probability Tree

modified to account for the structure of the random variable. Consider Figure 2-4(a) and Figure 2-4(b) [4].

Both probability trees represent the same random variable. However, in the modified probability tree, outcomes S_1 and S_2 is grouped into a subtree. The entropy formula can be re-written to

$$\begin{aligned}
H = &-[p_1 logp_i + p_2 logp_2 + p_3 logp_3] && (2.2)\\
= &-[(p_1+p_2)(\frac{p_1}{p_1+p_2} logp_1 + \frac{p_2}{p_1+p_2} logp_2) + p_3 logp_3] && (2.3)\\
= &-[(p_1+p_2)log(p_1+p_2) + p_3 logp_3] + (p_1+p_2)H_{12} && (2.4)\\
& && (2.5)
\end{aligned}$$

[4]Adopted from Information Theory for Systems Engineers[11]

where

$$H_{12} = -\left(\frac{p_1}{p_1+p_2} log \frac{p_1}{p_1+p_2} + \frac{p_2}{p_1+p_2} log \frac{p_2}{p_1+p_2}\right) \tag{2.6}$$

Equation 2.5 can be generalized to probability trees of arbitrary depth.

$$H_y = -\sum_{i=1}^{N} p_{x_i} log(p_{x_i}) + p_{x_i} H_{x_i} \tag{2.7}$$

where y is the root node of the sub-tree, x_i is one of the N immediate child node of the node y and p_{x_i} is the probability of outcome of the probability at y in the subset of outcomes represented by x_i. In summary, the entropy associated with the parent node (or the root node of the sub-tree) is the entropy of its immediate outcomes (the child nodes) and the weighted sum of the entropy associated with each child node, which themselves are root-nodes of their sub-trees beneath them.

This result can be used to measure the uncertainty of an ontology that is represented in a tree structure as described previously, and therefore the unpredictability of a user's interests at each moment in time.

2.3.2 Kullback-Leibler Distance

Related to the concept of entropy is *Kullback-Leibler distance*, or KL distance. KL distance (also known as *relative entropy*), $D(p||q)$, is a measure of the distance between two distributions.

$$D(p||q) = \sum_{i=1}^{N} p_{x_i} log \frac{p_{x_i}}{q_{x_i}} \tag{2.8}$$

KL distance, $D(p||q)$, measures the error in predicting the uncertainty of a random variable X when its probability distribution is assumed to be q while the true distribution of X is p. In other words, if the true distribution p is used, $H(p)$ is the uncertainty of X. However, if instead q is assumed, the resulting uncertainty of X is $H(p) + D(p||q)$. Note that $D(p||q)$ does not neccessary equal to $D(q||p)$ and it does not obey the triangle inequality of distance. As a result, it is not a true distance measure. Nonetheless, KL distance has useful properties:

- $D(p||p) = 0$
- $D(p||q) > 0 \; if \; p \neq q$

As such, KL distance is usually used as a measure of the deviation of $q(x)$ from $p(x)$.

The KL distance is composed of two components,

$$D(p||q) = -(\sum_{i=1}^{N} p_{x_i} log(q_{x_i}) - \sum_{i=1}^{N} p_{x_i} log(p_{x_i})) \tag{2.9}$$

The first component is the entropy of the random variable encoded with the probability distribution q, while the second component is the actual entropy of the random variable. Since KL distance is the linear superposition of the two components, the derivation of the recursive equation in Section 2.3.1 can be applied to KL distance. Equation 2.8 can be rewritten into

$$D_y = -\sum_{i=1}^{N} p_{x_i} log \frac{p_{x_i}}{q_{x_i}} + p_{x_i} D_{x_i} \tag{2.10}$$

2.4 KL Distance as a Similarity Measure

The previous sections on ontologies and Information Theory set the tone for the use of Kullback-Leibler distance as the measure of similarity between users.

Consider the usual representation of a user or search inputs, where each keyword F_i in the feature vector $(f_1, f_2, f_3 ...)$ is a feature the user is interested in. The keyword matching algorithm uses the degree of overlap of the keywords to represent the similarity. The cosine similarity measure (which is essential the inner product of two keyword vectors) is the extension of the basic keyword overlap to take into account non-binary weights of keyword vectors.

The keyword vectors can be re-formulated to use KL distance as the matching algorithm. The feature vector can be recast into a shallow probability tree where each of the features, F_i is an outcome to the root node that represents the user's interests (or the search input).

The probability associated with each edge can be regarded as a measure of the relative importance (or relevance) of the concept represented by each child node to each user. KL distance can then be found by using equation 2.8, where the user is represented by the probability distribution $p(x)$ while the other user (or document) is represented by $q(x)$.

The advantage of using KL distance over keyword overlap become clearer in general problems that involves multi-level probability trees. In such problems, the semantics of the keyword overlaps becomes unclear because it does not consider the relations between concepts. In contrast, KL distance remains obvious; it is simply the measure of the deviation between the probability distributions.

There is, however, a slight difference in interpretation of the tree structure described in Section 2.2 and Section 2.3. In probability trees described in Section 2.3, the use of multi-level probability trees is artificial; the KL distance is the same in a shallow probability tree and its multi-level form. In the treatment of the ontologies however, the probabilities assigned at each level in the tree are usually independent of the probability distributions at other different levels of the tree. The tree structure of the ontology can be flattened into a shallow tree, where the probabiltiy of each child node can be computed by multiplying all the probabilities in the path from the root node to same child node in the multi-level representation; the KL distance will still be the same in both representations.

This appears to be an argument in favor of using keyword overlap rather than KL distance. Afterall, if the multi-level probability trees can be flattened into a 1-level tree, there can also be an equivalent feature vector. However, it is important to keep in mind the implicit structure of the flattened tree; the keyword overlap algorithm ignores such structure in the tree. Therefore, when learning the user's interest profile, there are more advantages to use ontologies than feature vectors. While the semantics of keyword overlap is only clear in feature vectors (and the equivalent to the shallow tree), the interpretation of KL distance is the same for any arbitrary tree structure. I believe that tree-representations of the ontology are richer and more accurate descriptions of a user's interests, KL distance is therefore more general than keyword overlaps as a similarity measure.

For a function to qualify as a metric, it needs to satisfy the *positivity*, *symmetry* and *triangle inequality* requirements. KL distance, however, satisfies the positivity clause. As such, it is not a metric in its strictest mathematical sense. However, KL distance remains widely used as a *deviation*. This is further discussed in Chapter 4.

Chapter 3

Interest Matching in Online Communities

3.1 Learning the Interest Profile

Before interest matching can be automated, the system must have accurate descriptions of each user. The process of learning the descriptions is generally called user profiling. Within the context of interest matching, user profiling is the process of learning the interest profile of each user, or *personalizing the standard ontology.*

User profiling can be a complex process. In order to obtain an accurate representation of the user, the system must monitor the user activities in certain ways. Data mining techniques are then applied on the collected data to extract possible patterns inherent in user activities. Although this thesis does not focus on profiling users, the process of interest matching will not be complete without introducing ways of learning the interest profile.

Before exploring the usual methods of data collection, it is useful to differentiate between a *class* and an *instance.* A class represents the area of interest and is synonomous with a *concept.* An instance is a realization of a class. For example, in the Yahoo Ontology, *Art History* is a class, while websites that are classified in *Art History* are instances of the class. In the tree representation of an ontology described in the previous chapter, each node represents a class. Based on this differentation, ways of obtaining user data can be classified as follows:

Feedback Forms The structure of the ontology can be exposed to the user. Each user can then be asked to highlight their areas of interest and post the selections back to the system. For example, in Figure 2-1, the user can select nodes B and D, and the personalized ontology represents his or her interests.

Rating Systems Instead of asking each user to select the classes of interests, the system can ask the user to rate an instance of the class. For example, in the context of *Art History*, the user is asked to rate the articles that are under that genre. For each rating of approval, the weight of the class in the ontology can be increased to reflect increased relative importance. If the user gives a negative rating, the weight is decreased to reflect dislike. In general, if the user tends to rate instances highly, it is reasonable to assume that the user would be interested in the class to which the instances belong. This method of user profiling produces a better interest profile because the user does not have conscious control over his or her profile. The disadvantage is the time required to learn the user profile.

Access Patterns The two previous methods require explicit feedback from the user. However, in most cases, it is undesirable to interrupt the user from his activities to ask for feedback. Sites like Ebay [1] and Amazon, which require user feedback, face the problem of many users not taking the time to reflect his or her opinions on the item or area. As a result, the process of learning the user's interest profile can be prolonged and become less accurate. Less invasive, or implicit techniques of user profiling have been developed to circumvent this problem [4]. The system can observe the user's activities and the observations can then be used to estimate the user's level of interest in instances through the use of heuristics. For example, in learning a user's interests in various categories of websites, if the user accesses the same sites regularly and spends a copious

[1] http://www.ebay.com

amount of time browsing through the site, the system can safely assume that the user has a higher relative interest for the category to which the websites belong, compared to other categories whose the sites are accessed less frequently.

In the latter two methods, there may be a danger in assuming that if the user shows interests in instances of the class, the user is likely to be interested in that class itself. After all, there can be a wide variation of quality in the websites that are classified under Art History. On the other hand, if there are a wide disagreement about the quality of the instances, it is likely that the class requires further decomposition into sub-classes to which the instances are to be reclassified. [2]

Although one may use a variant of the above methods to gauge user interests, the resultant interest profile learnt is independent of the method of data collection. Since it is the aim of this thesis to compare profiles rather than to learn profiles, unless stated otherwise, the interest profiles used in the experiments can be assumed to be obtained by using feedback forms.

3.2 Sensitivity Analysis

This thesis presents an analysis of a number of simulations exploring the ideas described. The goal of the simulations is to test the validity of the ontology-based approach. A sensitivity analysis is performed to compare the behavior the ontology-based algorithm with the keyword-overlap algorithm (specifically the cosine similarity algorithm).

The ontology used in the simulations is the concept hierarchy of the CenterWatch Clinical Trials [3]. It is a classification scheme for clinical trials all over the United States, where each clinical trial classified into various fields of medical studies.

The ontology is a singly-connected tree with a depth of 3. It contains 6000 nodes, and each node is assigned a unique id. Each node has an associated weight attribute that indicates the relative importance of the node with respect to other nodes. Each

[2]For more examples of user profiling, please refer to [15]
[3]http:www.centerwatch.com

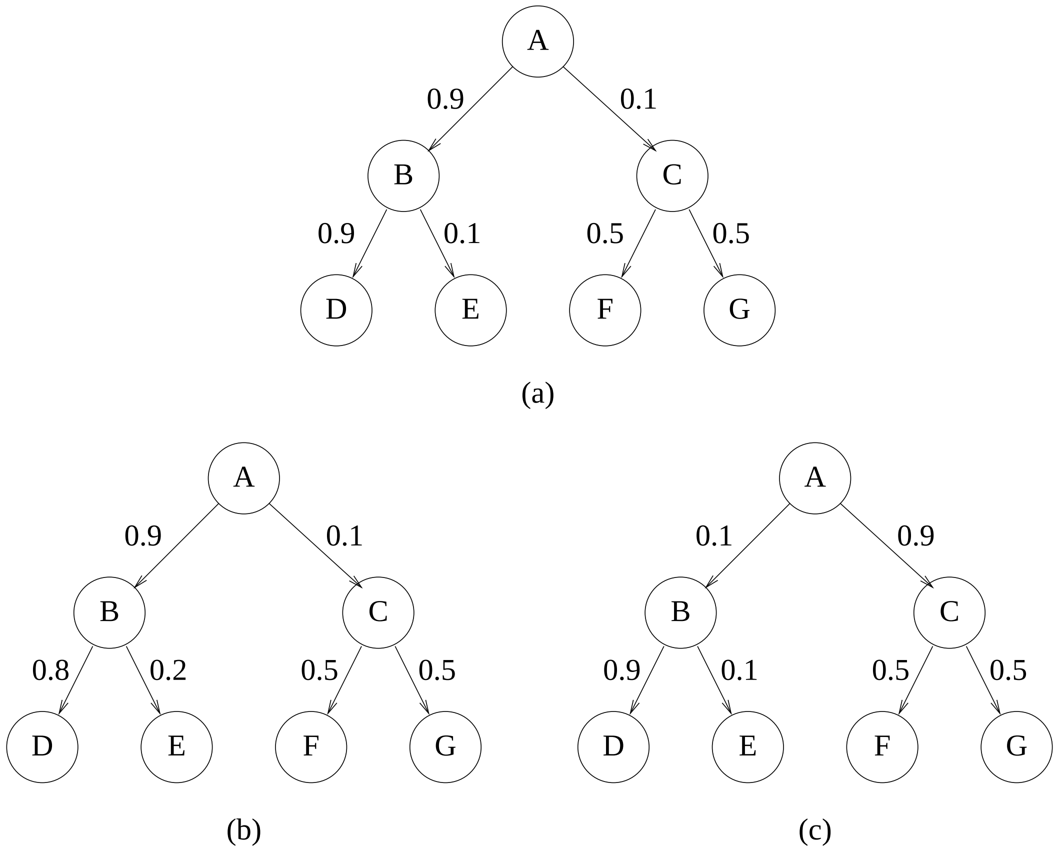


Figure 3-1: Interests Profile of (a) A (b) B (c) C

node starts with a small default value. This ontology is adopted by an imaginary user of the system. For each user, to simulate interest in a particular area (node), the weight of the node is raised to a stronger number so that when compared to another node with the default weight, the simulated user has a 90% chance of picking the node of interest over the unhighlighted node.

3.2.1 Simple Cases

Consider three users A, B, C in the network that has adopted a common ontology as shown in Figure 3-1. Each of them has highlighted certain aspects of the ontology that they are interested in. The probabilities associated with each edge are the weights assigned to the corresponding child node normalized with respect to the sum of the weights of all its sibling nodes and itself.

On first look, it is intuitive to say that users A and B are more similar to each other compared to user C. After all, they only have a slight disagreement over the subtree node B as to how much they like D and E. User C, however, shows a strong bias for node C. The ontology-based algorithm takes account of such multi-level classifications and relations between each node. The resultant measures are:

$$D(A||B) = 0.033$$
$$D(A||C) = 1.759$$

The smaller the KL distance, the more similar two users are. Therefore, in comparing users A, B and C, the algorithm predicts that user A and user B are more similar to each other than user C is.

The more interesting problem is to rank the community in terms of similarity with respect to user C. Which of user A or user B is more similar to user C?

$$D(C||A) = 1.759$$
$$D(C||B) = 1.761$$

The algorithm determines that user A is more similar to user C than user B. Although it is not immediately obvious, the subtree of node B of user C is similar to user A's interest profile. Although the keyword-overlap algorithm should give the same indication, an ontology offers a better insight as to how users are similar (or dissimilar). It allow an observer to focus on the section of the two interest profile that generates in the increase of the similarity measure.

3.2.2 Mutations

To observe how the ontology-based algorithm is able to cluster users into similar users compared to the keyword-overlap algorithm, consider two users who have the exact same interest profile with a small subset of nodes of the ontology highlighted as common interests (10% of the total number of nodes). Each interest profile is allowed to evolve independently by subjecting it to a *mutation* at every time step.

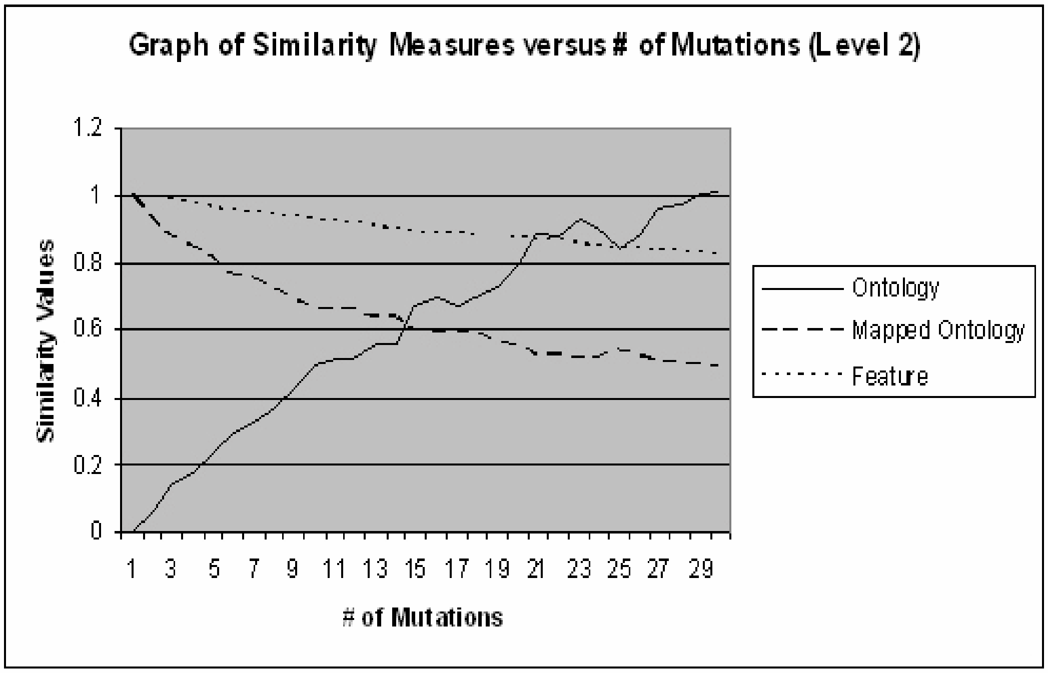


Figure 3-2: Mutations at Depth of 2

A mutation is the act of adding or removing interests to a user's profile. Adding or removing interests occur with equal probability of 0.5.

The keyword-overlap algorithm is expected to decrease from 1 in a hyperbolic manner as it is the simple normalized count of the decreasing coincidental interests. In Chapter 2, I have argued that a feature vector is essentially a shallow single-level tree. Therefore, If mutations are constrained to a single level of the tree, the behavior of the ontology-based algorithm should be similar to the feature-vector algorithm.

Constraining to Single Levels of Mutation

As expected, the general trends of both the ontology-based and feature vector-based algorithm exhibit similar characteristics when mutations are constained to one level of the ontology. Unlike the smooth curve of the feature-vector based algorithm, the graph of the ontology-based algorithm, however, exhibits some randomness about the general trendline. This is due to the fact that although mutations are constrained

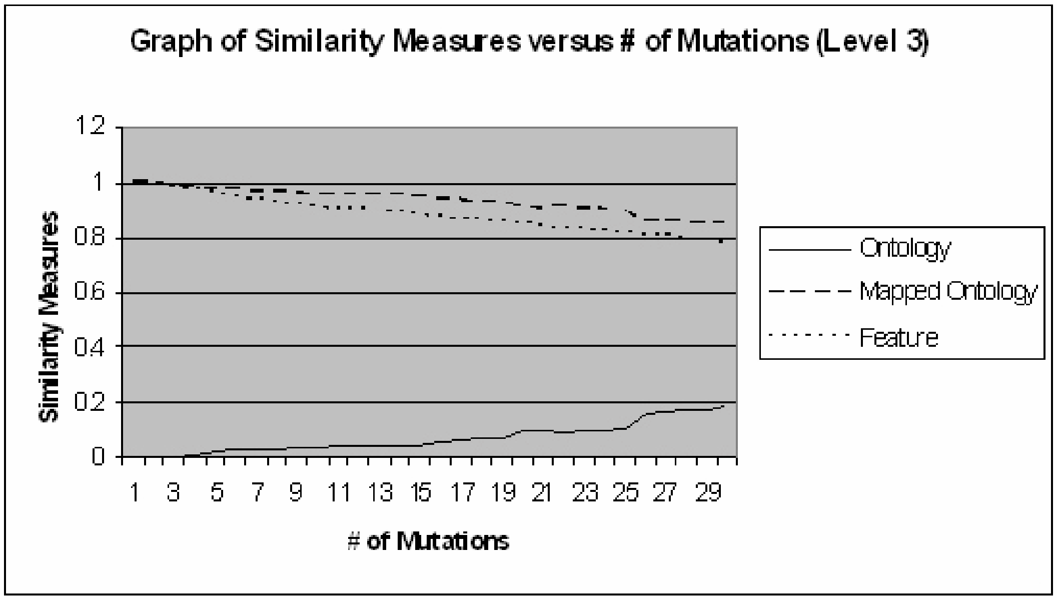


Figure 3-3: Mutations at Depth of 3

occur at the same level, they can occur in different sub-trees of the ontology.

Figure 3-4(a) shows the part of the ontology before mutation. Figure 3-4(b) and (c) are different possible scenerios for mutations to occur while being constrained to the same tree level. In Figure 3-4(b), the mutation occurs in the same subtree as the original area of interest was. In Figure 3-4(c), the mutation occurs in another subtree of the ontology at the same level. Based on the description in Chapter 2, the new ontology in (b) remains more similar to (a) than (c) is to (a). The lack of smoothness of the curve is a result of these considerations.

Another interesting point in Figure 3-2 in how the rate of change in the ontology-based algorithm differs when mutations are constrained at different levels of the ontology. In Figure 3-2(a), where mutations are allowed only at level 2, the rate at which the two users become dissimilar is considerably faster than in Figure 3-2(c) where mutations are only allowed in level 3. Note that the feature-vector based algorithm exhibits the same behavior in both constraints. It is a direct consequence of the tree-based structure of the ontology. From a social standpoint, this is useful because although two users may differ in their preferences, the deeper in the ontology

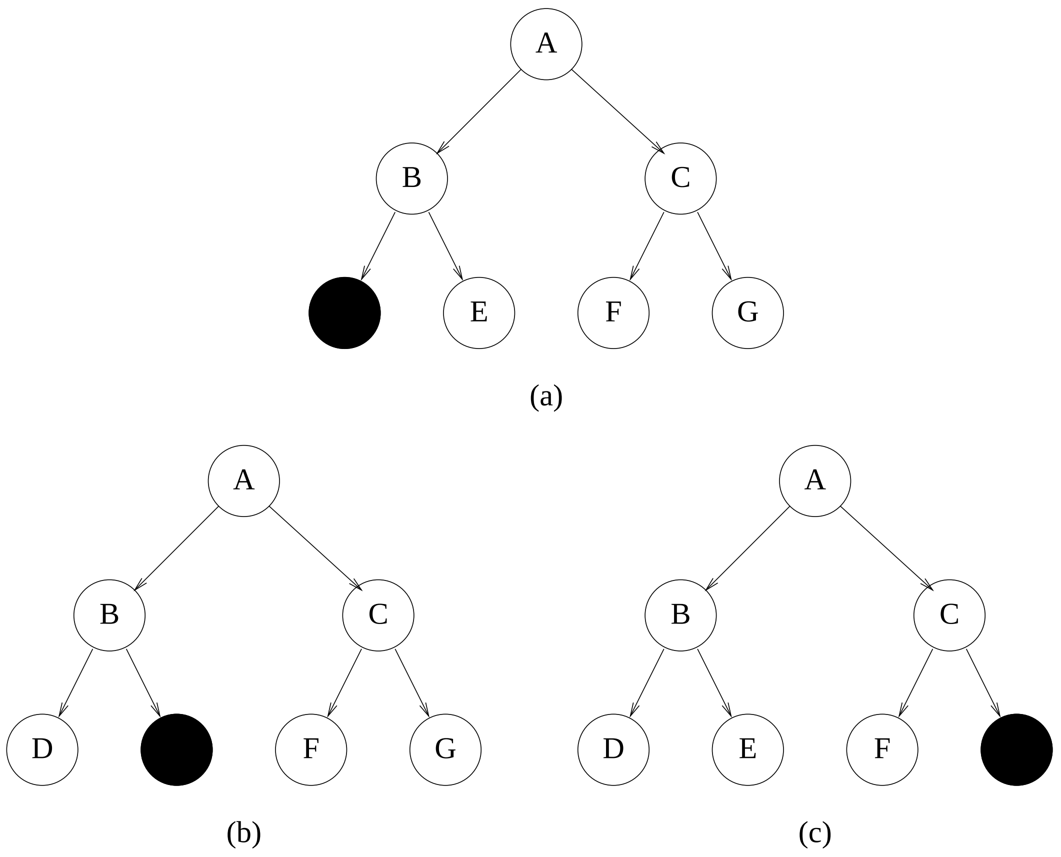


Figure 3-4: Different Mutations

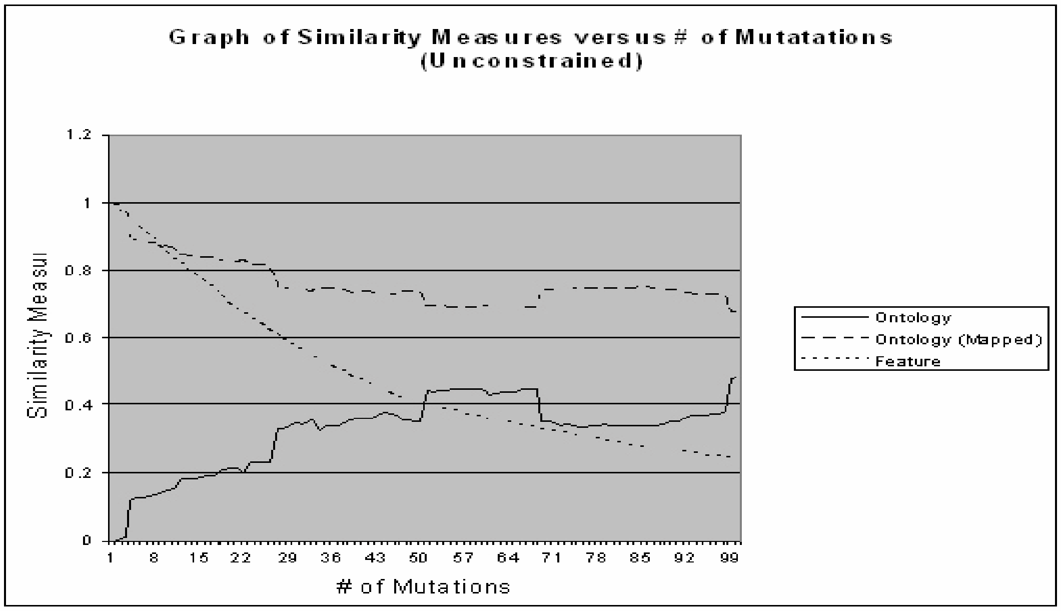


Figure 3-5: Graph of Unconstrained Mutations

tree the differences occur, the less dissimilar they are. After all, if two users like *cars* while one prefers *Benz* and the other *Porsche*, even if their interests changes within the sub-tree with the root node being *cars*, they are more similar to each other as compared to another user who prefers *food* to *cars*, both being siblings.

Unconstrained Mutations

Figure 3-5 shows the results when the mutations are allows to occur at any point in the tree. The graph of the ontology-based algorithm shows strong jumps at certain points of teh curve while the curve of the feature-vector algorithm remains smooth. The cause of the randomness is similar to the explanation presented in the previous section on mutations migrating between subtrees. However, in this case, mutations can migrate between levels of the tree (shown in Figure 3-6). This causes the violent swings in the curve.

An interesting portion of the ontology-based curve in Figure 3-5 is the region around the origin, before the sudden increase in the gradient that. Although there has been some mutations, the resultant mutated interest profile remains relatively

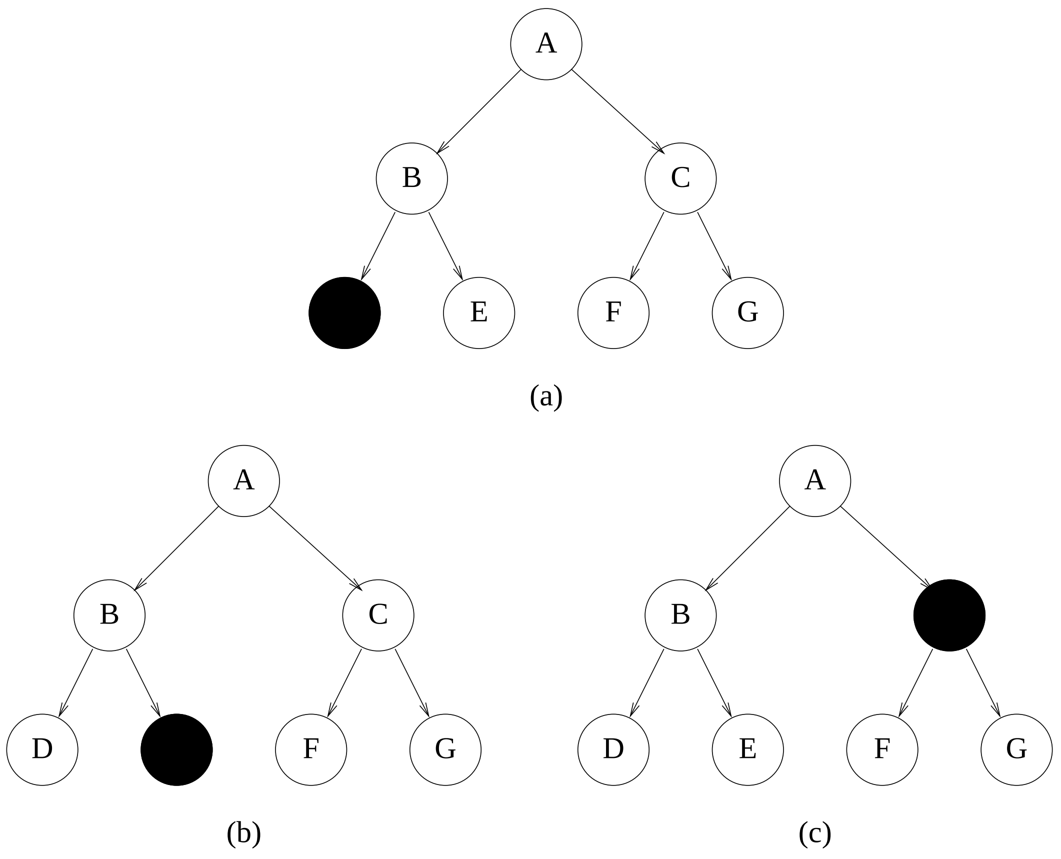


Figure 3-6: Different Mutations

similar to each other, compared to the later portions of the curve.

The provides a useful heuristic for clustering users in a community. In a large community, any interest profile can occur with an equal probability as any other ontologies. All the interest profiles in the community can be ranked in ascending dissimilarity with respect to a single interest profile using the ontology-based algorithm. The graph of the resultant measures are likely to be similar to Figure 3-5 with violent upswings at certain points of the curve. All the users ordered before the first upswing can be considered to be the most similar to a particular user in the whole community. This clearly has significant advantages over the feature-vector approach. Since the feature-vector approach will result in a smooth graduation from the point of maximal similarity, it is difficult to define a cut off point between users who are similar and dissimilar to any particular user.

3.2.3 Noise Tolerance

A useful matching algorithm must be able to tolerate noise in the probabilities of the interest profile. It is likely that the user profiling process is not 100% accurate and has to be refined over time. This is especially true in profiling the user through ratings and background observations. Therefore it is desirable to compare the accuracy of both the ontology-based and feature-vector based algorithms.

Consider the CenterWatch ontology that is a classification of the clinical trials in the United States. There are three interest profiles: A, B and C. They have the same exact weights on each node at time 0. Each node has a default weight of 10. Interest profile A is assumed to be static. Interest profile B and C has a randomly selected subset of nodes whose weights vary according to a Gaussian distribution, thus simulating noise. The subset is constant over the entire process of simulation. For B, the distribution of the weights have a mean of 10, while the distribution for each of the selected weights has a mean of 9. The standard deviation of all the Gaussian distributions is varied from 0 to a standard deviation of 2, which is twice the difference of the two means.

For ontology-based algorithm, the sum of the weights of all the siblings that

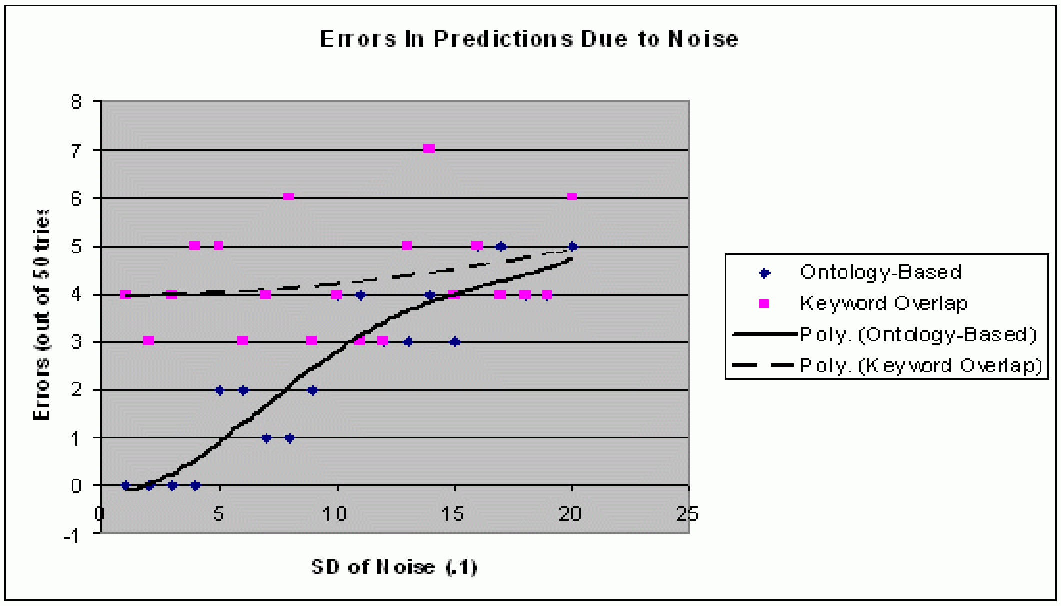


Figure 3-7: Graph of Errors versus Noise Strength

share that common parent node is normalized to 1 so as to recast the interest profile into a decision tree. For the keyword-vector based algorithm, all the nodes in the CenterWatch ontology are flattened into a keyword vector.

For a series of steps, the standard deviation is increased by a small amount to reflect an increased amount of noise in the system. All the nodes in the selected subset for B and C are resampled to obtain new weights. The new interest profiles B and C are then compared to interest profile A using both the ontology-based algorithm and the keyword-vector algorithm. A *mismatch* occurs when the algorithm indicates that C is more similar to A than B is to A. For each step, a number resampling of the weights of all the nodes is performed to find the average number of mismatch each algorithm is likely to make.

Based on the previous description, it is expected that the matching algorithm would say that interest profile B is more similar to A than C is to A.

3.3 Interest Matching in Online Communities

Interest matching can be deployed in online communities in a variety of ways. In peer-to-peer networks, users tend to have resources that similar uses would be interested in. As such, a challenge in online communities would be to locate similar users in the shortest amount of time. Interest matching can also be applied as a heuristic to locate similar individuals.

Consider a community of entities, each containing an interest profile. The goal of each entity is to locate the top two most similar agents to itself in the network. To model peer-to-peer networks and most offline communities, each entity starts off by knowing a small number of entities in the community. When simulation proceeds, each entity proceeds to contact other entities it knows in the network to learn other agents in the network previously unknown. In the simulation system, each agent is assigned a small ontology with weights, varying from 1 to 100, randomly assignd to each node in the ontology. The similarity measure is the KL distance adopted in this thesis.

When an entity receives a request for new contacts, there are two strategies for selecting a subset of agents to return,

- random - it selects a random subset of entities to return to the requesting entity
- best - it returns the subset of most similar entities to the the requesting entity

The number of messages passed by each entity before the top two entities indentified remain unchanged is used as an estimate of the efficiency of the strategy adopted; the smaller the number of messages passed before the top two users stabilize, the better the stratregy adopted. Clearly, each user's time-to-stability is different. As such, the average time-to-stability is computed for all the entities' in the network.

Figure 3-8 shows the graph of the average number of messages before the top two entities are finalized versus the number of entities in the network. The trend in both strategies follows a logarithmic dependence with respect to the number of agents in the network. Because of the limited number of contacts that an entity knows at each

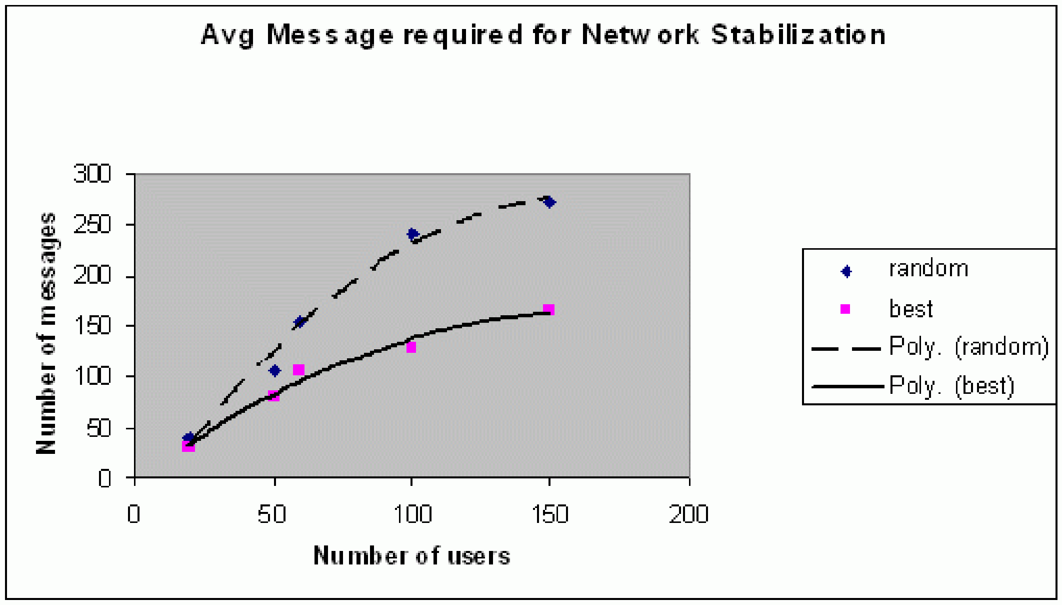
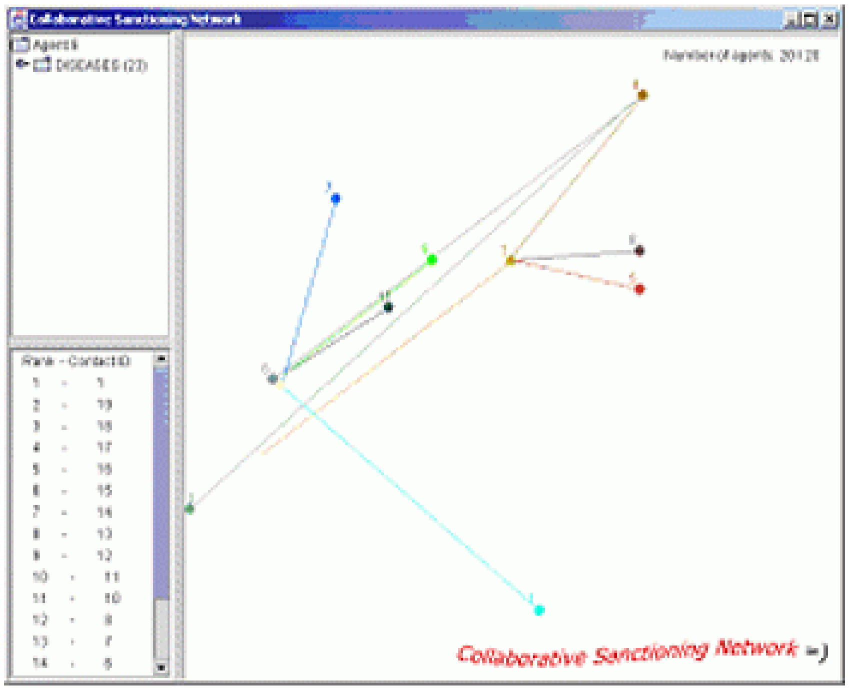


Figure 3-8: Graph of Number of Messages versus Number of Agents

moment in time, it essentially traverses the network of entities in a manner akin to traversing a tree. As shown in the graph, the *best* strategy clearly reduces the average number of messages by half when compared to the *random* strategy.

3.3.1 The Collaborative Sanctioning Network

The simulation system described in the previous section shows that the similarity can be used as a heurisitic in locating other entities more efficiently. Preliminary work in deploying the interest-matching algorithm has taken the form of agents in the peer-to-peer network. Each agent is a program connected to the network and is capable of automated discovery of other agents in the network. In addition to discovery protocol, agents can communicate with other agents in the network via message passing. In the context of our research, the network of agent is simulated in a prototype environment called Collaborative Sanctioning Network (or CSNet). In addition to discovery protocols, each agent also contains an ontology that represents the user's interest. The program strucure is descibed in Appendix A.


Collaborative Sanctioning Network
Agent 6
DISEASES (23)
Number of agents: 20 / 20
Collaborative Sanctioning Network =)

```
Rank - Contact ID
  1  -   1
  2  -  19
  3  -  18
  4  -  17
  5  -  16
  6  -  15
  7  -  14
  8  -  13
  9  -  12
  10  -  11
  11  -  10
  12  -   8
  13  -   7
  14  -   6
```

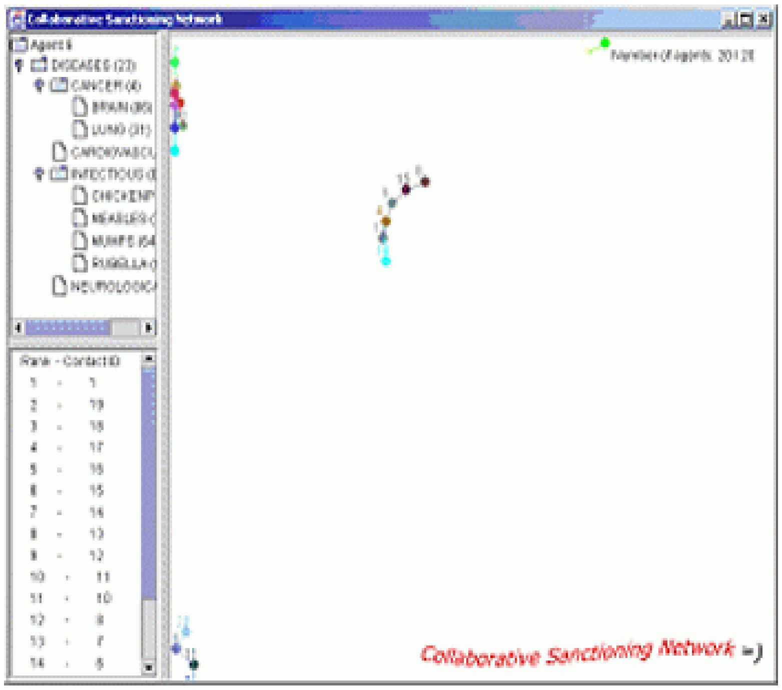
Figure 3-9: Initial display

Results

Each entity in the network is comprised of the three inter-connecting modules. The goal of each entity is to find out which entities in the network are most similar to itself. This is achieved by message passing. To observe how the system evolves, a graphical overlay is written over the network of entities.

Each entity is represented as a dot and placed on the main display as it is created in the network. The upper left segment shows the interest profile of each entity when selected, while the lower left segment shows which entities in the network it has contacted and their present ranking with respect to itself. Figure 3-9 is a snapshot of the image at the time near program initialization.

As an entity starts to contact other entities, it will have a ranking of all the entities it knows at any moment in time. To accentuate the rankings, each entity will pull the highest ranked entity (that it knows at that point in time) in the network towards itself (in terms of the graphical distance). To prevent the eventual collapse of every



```
Rank - Contact ID
 1  -  1
 2  -  19
 3  -  18
 4  -  17
 5  -  16
 6  -  15
 7  -  14
 8  -  13
 9  -  12
 10 -  11
 11 -  10
 12 -  8
 13 -  7
 14 -  5
```

Figure 3-10: Final display

entity to a single point, each entity will also push the lowest ranked agent (that it knows at that point in time) away from it. The system is allowed to evolve until the point where the relative positions of each entity remains relatively unchanged. Figure 3-10 shows a snapshot of the final state of the graphical representation of the entities in the network.

In Figure 3-10, out of the 20 entities in the network, several clusters of entities can be delimited. Each cluster is the maximum inclusion of the entities that has ranked another entity in the network as being most similar to itself. Since each entity want not preclassified, one can interpret each cluster to represent a *dynamic* context in which each entity can be classified into. This could allow systems, such as reputation management and dynamic chatroom, to be built on top of the concept of *dyamic clusters* and *contexts*.

The graphical display introduces a controversy into the interpretation of the *clusters* and the validity of the relative positions of the clusters. See section 4.2, for further discussions.

Chapter 4

Discussions

This chapter focuses on the implications of using the Kullback-Leiber distance as a similarity measure and further discussions of the experimental results obtained.

4.1 Kullback Leibler distance as a metric

Kullback-Leiber distance is a measure of the deviation between two probabilities distribution. In this thesis, each user is represented by a probability distrubution over a random variable. The outcomes to that random variable is the same for each user. A metric must satisfy the three following properties:

1. positivity: $0 < f(x,y) < \infty$, $x \neq y$ *and* $f(x,x) = 0$
2. symmetry: $f(x,y) = f(y,x)$
3. triangular inequality: $f(x,y) \leq f(x,z) + f(z,y)$

where f is a function that maps x and y to a metric space.

KL distance satisfies neither the symmetry clause nor the triangular inequality. As a result, it does not represent a true metric distance between the two distribution. However, it does satisfy the positivity clause. As a result, in many fields of application, it is regarded as a measure of *deviation* between p and q.

In Section 3.3.1, a graphical representation of the entities is described. The creation of the clusters raises questions of the validity of the use of KL distance.

Implications of lack of symmetry to an external observer

Since the distance between users is not symmetrical, a spatial representation of the *distances* between the users may not be meaningful. However, it is important to realize that the approach in this thesis is to adopt the viewpoint of a user. Rankings are performed with respect to a user. Therefore, the KL distance is only meaningful with respect to that user. As such, symmetry is not a strict requirement.

In the process of developing this thesis, other approaches had also been considered.

Symmetric KL distance A simple way to circumvent the problem of symmetry is to use D, where

$$D = D(p||q) + D(q||p)$$

However, in using the measure, the semantics of the symmetric KL distance becomes unclear. The symmetry measure is viewed as an accuracy measure of how well another user's interest profile can be used to approximate the user in question. According to the definition of similarity in Chapter 2, similarity is defined to be the resulting error is an interest profile belonging to user B is used to describe user A. The symmetric KL distance includes the deviations from the perspectives from both users. Although the resulting quantity is symmetric, it is no longer clear what the quantity actually represents.

Mutual Information Mutual Information is the KL distance between the joint probability distribution and the product of each of the probability distributions. The formula is given by

$$I(x:y) = \sum_{x}\sum_{y} p(x,y) log \frac{p(x,y)}{p(x)p(y)} \tag{4.1}$$

$$= D(p(x,y)||p(x)p(y)) \tag{4.2}$$

Mutual Information, I, is a measure of the amount of information in contained in random variable x about random variable y. It satisfies both the positivity and symmetry clause of measures. In the context of interest matching, each user can be modelled as a random variable. $p(x)$ and $p(y)$ are therefore different probability distributions over the same outcomes. Constructing the probability distributions can be learnt from the methods of user profiling as described in Chapter 3. $p(x, y)$ is the joint probability distribution of two random variables. Learning the joint probability distributions of two users requires the system to track pair-wise user behaviors in the system. For example, for a particular outcome o_i, $p(x = o_i, y = o_i)$ is the probability that both users shows strong interest in the particular outcome.

In summary, Mutual Information is a measure of the dependence between the two random variable. If there is complete independence, that is $p(x, y) = p(x)p(y)$, the Mutual Information reduces to 0. However, in the application of interest matching, it appears to be unreasonable to compute dependencies between users. After all, if users have never interacted with each other before, there should not be any dependence of interests on other users in the network.

4.2 On the Collabative Sanctioning Network

Figure 3-10 shows a graphical interpretation of the final state in the evolution of the agent network. Each *cluster* (in the graphical sense) comprises of a group of agents that are deemed to be *similar*.

As described in the previous section, KL distance does not qualify as a distance metric. As such, the interpretation of the cluster in the graphical display would appear to be ambiguous. If the distance between each node (representing an agent) is assumed to represent the similarity between the agents, it will not be meaningful to consider the pixel distance between clusters.

However, it is important to consider how each cluster of agents is formed. As described in Chapter 3, each agent endeavors to discover which other agent in the network is most similar to itself. Having located another agent that is most similar to the agent in comparison to all the other agent it presently knows, the distance between the two agents will start to decrease in the graphical display gradually to a small range of values. A line is also drawn from the agent to the other agent to represent the similarity (This is more clearly shown in Figure 3-9). In the final state, each cluster is a group of agents where each agent in the group is indicated by one or more other agent (in the group) to be most similar to itself (themselves). It is important to note that the graphical display is, by no means, a demonstration of KL distance as a metric. There is no meaning to the distance between the clusters; it is a simple by-product to accentuate the distinction between the clusters. Returning to a graphical metaphor, a cluster of agents are a group of agents that all linked together by lines. Another agent is considered to be outside the cluster if there is no line connecting it to another agent in the cluster. Therefore, the graphical display can regarded as a simple way to conceptualize the resulting network if an agent chooses to associate itself with its most similar agent in the network.

Chapter 5

Conclusions and Future Works

This thesis presented a novel approach to interest matching in online communities that is based on the combined use of an ontological description of each user's interests and the Kullback-Leibler distance as a similarity measure.

In order to perform interest matching, each user adopts a standard ontology sanctioned by the community. The ontology is then personalized through user profiling techniques. Every edge in the ontology becomes associated with a probability that captures the likelihood of the user's interest given a context. The personalized ontology is also called an interest profile. The similarity measure is defined to be the deviation between two interest profiles and is given by the Kullback-Leibler distance between them.

By imposing a hierachical classification, the sensivity analysis of the ontology-based interest matching algorithm displays comparatively more interesting behvavior in comparisons to the cosine similarity measures. This allows for better classification of the users in the online community that would not have been possible in the cosine similarity measure. Moreover, in modelling noise (such as errors in user profiling techniques or interest changes) in the interest profiles, the ontology-based matching algorithm has a stronger noise tolerance compared to the cosine similarity measures. The similarity measure derived in this thesis is shown to be a useful heuristic in discovering other similar users in the community in an efficient manner. Requesting users presently known to return new contacts most similar to themselves increases the probability that a user can locate other users most similar to him or her self.

Future work will focus on extending the theory of the ontology-based interest matching algorithm. In this thesis, the ontology adopted is assumed to be a singly-connected tree. This, clearly, is not representative of general ontologies, which are often represented as polytrees. This brings about multiple dependences in the tree structure. It is likely that Bayesian Theory would be useful in describing such dependencies. As a side note, present approaches to interest matching and information retrieval focus upon learning the weights (or probabilities) associated with each concept. It would be interesting to extend user profiling techniques to learn the structure of the ontology for each user. The challenge then lies in matching ontologies with the same nodes but different structural relations.

Appendix A

Program Structure

The structure of the Collaborative Sanctioning Network is described in this appendix. Each agent program can be separated into three distinct modules: Profile, ReqHandler and Scheduler.

A.1 Profile

The profile module encapsulates information about the user and whom the user knows in the network. Running the Profile module requires two input files that contain configuration information pertaining to the user (personal.xml) and the neighbors in the network that the user may know (neighbors.xml). The configuration files are described in XML syntax, each accompanied by a corresponding DTD.

The personal.xml file contains the ID of the user, the NET_ADDRESS (itself broken down into the HOST name and the PORT number) and the EMAIL address of the user. In addition, it also contains the ontological description of the user's interest and the last DATE when the ONTOLOGY was modified. The ontology is also described in XML syntax.

The neighbors.xml file contains a NEIGHBOR_SET, itself containing a collection of neighbors the user knows. Each NEIGHBOR also contains an ID, NET_ADDRESS and EMAIL of the neighbor. In addition to the administrative information of the neighbor, each NEIGHBOR also contains a CONTEXT_SET, itself containing a col-

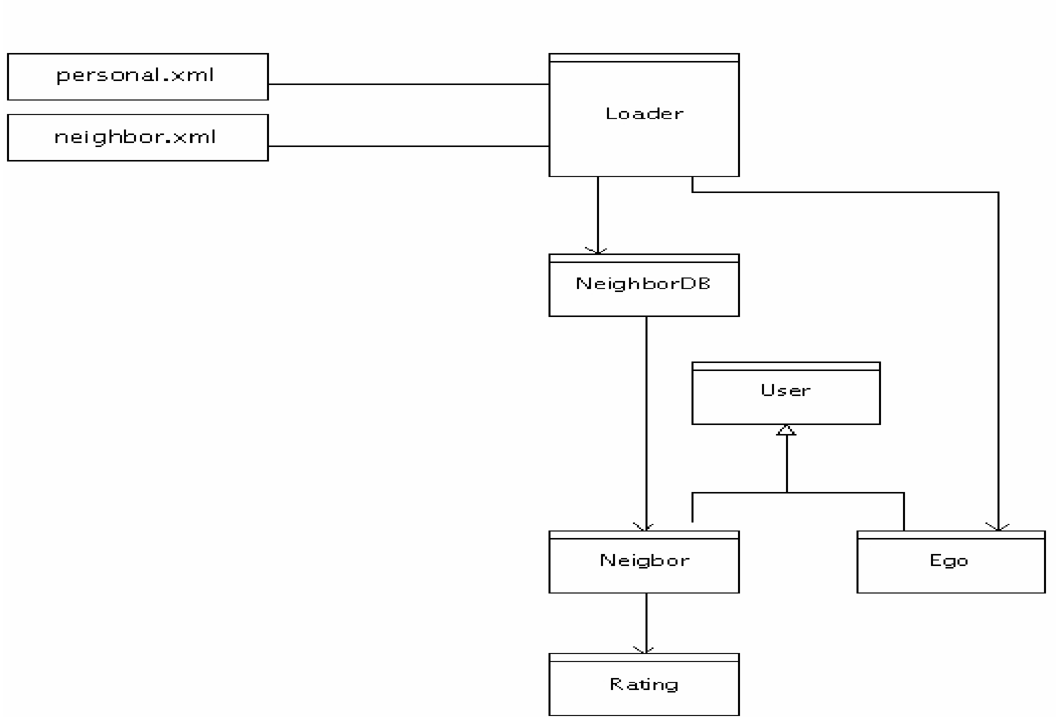


Figure A-1: MDD for ReqHandler module

lection of CONTEXT indicating subsets of the interests the user and the neighbor has compared each other. Each CONTEXT is a sub-tree of the ONTOLOGY of the user. Associated with each CONTEXT is a RATING that encapsulates measurements regarding the CONTEXT. For the purpose of interest-matching, the RATING contains a DATE as to when was the last comparison made and MEASURE, which contains the result of the Interest-Matching algorithm described in this paper. On program execution, these files are loaded into Java objects (specifically Ego and NeighborDB, objects representing personal.xml and neighbors.xml respectively) for other modules in the system. Figure A-1 shows the Module Dependency Diagram (MDD) for the Profile module [10].

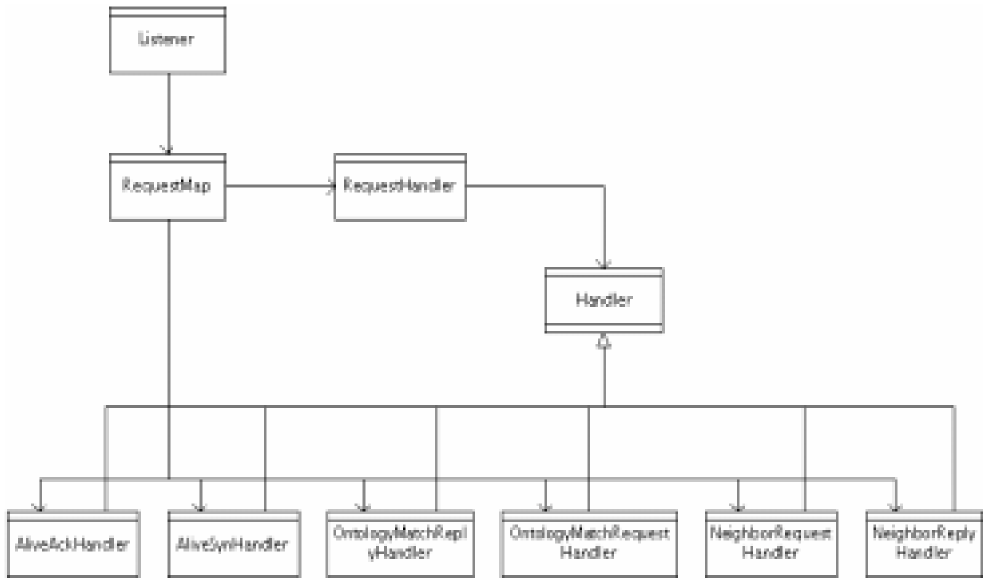


Figure A-2: MDD for ReqHandler module

A.2 ReqHandler

The RequestHandler module (or ReqHandler) consists of a set of handlers to handle incoming messages the agent may receive during operation. The Figure A-2 shows the MDD of the ReqHandler module.

Upon starting up the ReqHandler module, the Listener daemon starts listening to the network at the Port specified by personal.xml (accessed through Ego in the Profile module). Note that implicit in this MDD is that each object may have access to the Ego and NeighborDB instance during program execution. Upon receiving a request from the network, the message is passed to the RequestMap, which extracts the header from the message to determine which type of handler should be created to handle the message. There is a one-to-one correspondence to the message type and handler type.

From the Fgure A-2 above, one can observe six different kinds of Handlers in the system, each corresponding to the six messages existing at the moment this paper is written. Additional messages and its corresponding handler can be added into the MDD without disrupting the structure of the MDD. The only contract is that the

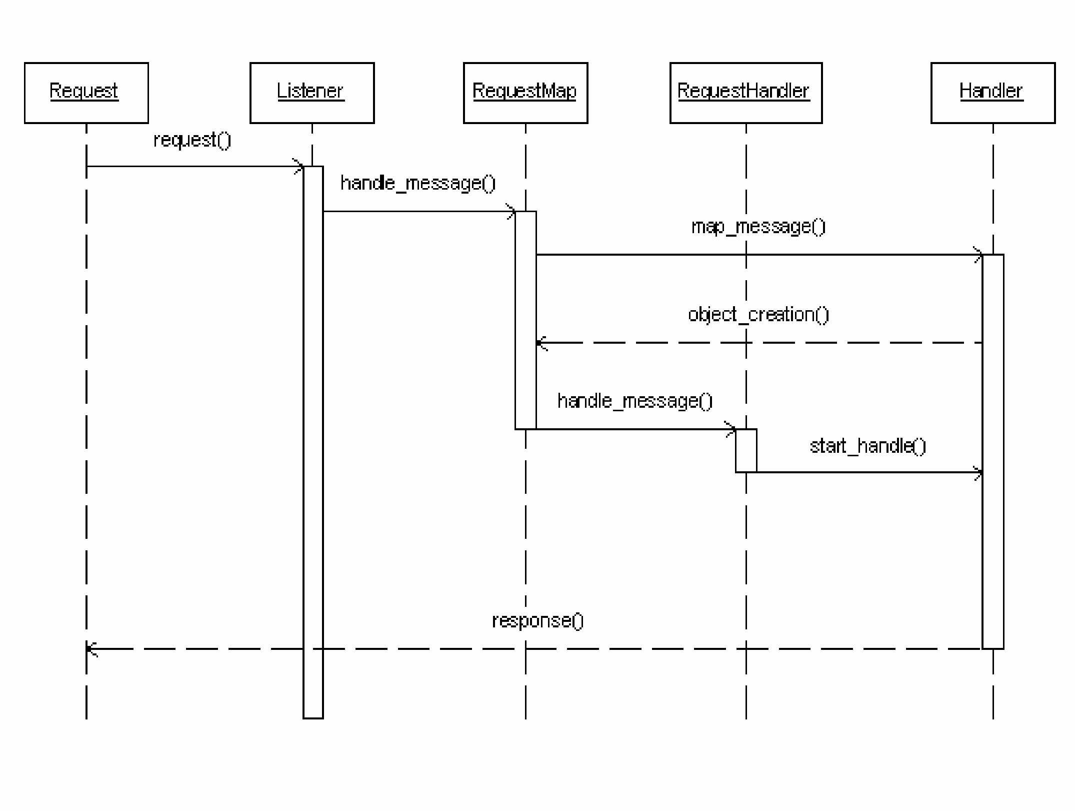


Figure A-3: Activation Diagram for receiving messages

Handler Name	Role
AliveSynHandler	1) adds sender to neighbors known by user 2) replies to sender with IS_ALIVE_ACK
AliveAckHandler	1) adds sender to neighbors known by user
OntologyMatchRequestHandler	1) performs ontology matching request 2) sends ONT_MATCH_REPLY to sender with results 3) adds results to its own neighbor database
OntologyMatchReplyHandler	1) adds results to its own neighbor database
NeighborRequestHandler	1) replies to sender with information regarding neighbor it knows
NeighborReplyHandler	1) adds the new neighbors sent to its own collection of neighbors

Table A.1: Table of Messages and Message Handlers

new handler must extend the Handler super class. The RequestMap (the handler factory) must also be modified to include the existence of the new handler. Table A.1 shows the message types and the corresponding handler and its role.

A.3 Scheduler

The ReqHandler module is essentially a set of message handlers. The Scheduler modules, in comparison, can be regarded as a set of schedules that are started up during program initiation. The role of each scheduler to start the process of searching for new contacts or to ask each contact to perform the interest matching algorithm with respect to itself. It's MDD has the same structure as that of the ReqHandler module, and is shown in Figure A-4.

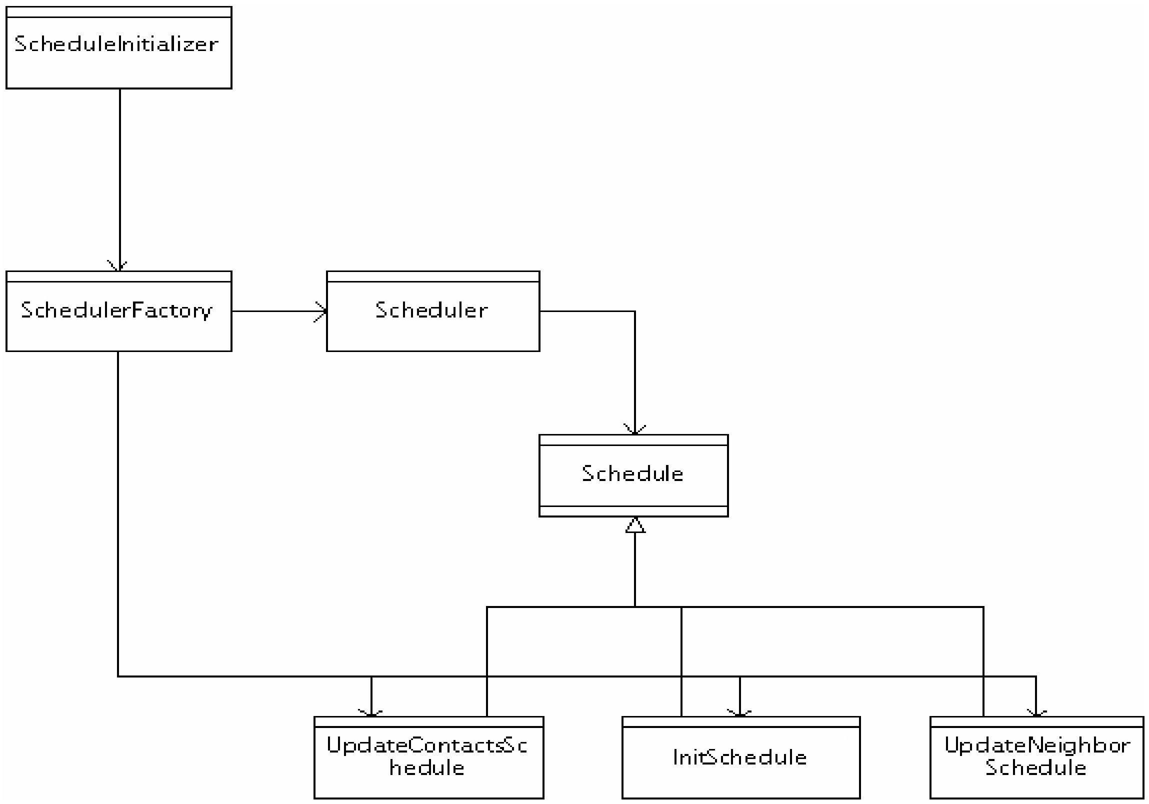


Figure A-4: MDD for the Scheduler Module

Bibliography

[1] Amazon recommendation service.

[2] M. Balabanovic and Y. Sholam. Fab: Content-based, collaborative recommedation. *Communications of the ACM*, 1997.

[3] F. Bellifemine, G. Poggi, and A. Jade. A fipa-compliant agent framework. In *University of Maryland*, 1995.

[4] P. K. Chan. A non-invasive approach to building web user profiles. In *KDD-99 Workshop on Web Usage Analysis and User Profiling*, 1999.

[5] T. M. Cover and J. A. Thomas. *Elements of Information Theory.* Jon Wiley & Sons, Inc, 1993.

[6] T. Finn, Y. Labrou, and J. Mayfield. Kqml as an agent communication language. Technical report, Proceedings of the 4th Internation Conference and Exhibition on the Practical Application of Intelligent Agents and Multi-Agents, 1999.

[7] L. Foner. Yenta: A multi-agent, referral-based matchmaking system. In *In Proceedings of Autonomous Agents Conference*, 1997.

[8] P. Greenspun. *Philip and Alex's Guide to Web Publishing.* Morgan Kaufmann, 1999.

[9] T. Gruber. A translation approach to portable ontology specifications. *Knowledge Acqusition*, 1993.

[10] J. Guttag and B. Liskov. *Program Development in Java: Abstraction, Specification, and Object-Oriented Design.* Addison-Wesley, 2000.

[11] L. P. Hyvarinen. *Information Theory for Systems Engineers.* Springer-Verlag New, 1970.

[12] L. Mui, M. Mohtashemi, C. Ang, P. Szolovits, and A. Halberstadt. Collaborative sanctioning: Enabled reliability and reputation ratings for distributed systems. Technical report, MIT Laboratory for Computer Science, 2001.

[13] Pointcast.

[14] A. Pretschner. Ontology based personalized search. Master's thesis, University of Kansas, 1998.

[15] A. Pretschner and S. Gauch. Personalization on the web. Technical report, Information and Telecommunication Technology Center, Department of Electrical Engineering and Computer Science, The University of Kansas, 1999.

[16] P. Resnick, N. I. Suchak, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work*, 1994.

[17] K. Sycara, J. Lu, M. Klusch, and S. Widoff. Matchmaking among heterogeneous agents on the internet. In *Proceedings AAAI Spring Symposium on Intelligent Agents in Cyberspace*, 1999.

[18] C. J. VanRijsbergen. *Information Retrieval.* 2nd ed., Butterworths, 1979.