

INFORMATION RETRIEVAL SYSTEM USING QUERY DISTANCE AND AGENTS

A DISSERTATION

*Submitted in partial fulfillment of the
requirements for the award of the degree*

of

MASTER OF TECHNOLOGY

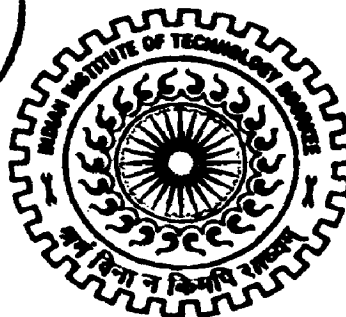
in

ELECTRONICS AND COMPUTER ENGINEERING

(With Specialization in Computer Science Engineering)

By

ARUNA KUMARI VAJJAGIRI



**DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE**

ROORKEE -247 667 (INDIA)

JUNE, 2008

Student declaration

I hereby declare that the work that is being presented in this dissertation report entitled “**Information Retrieval System using Query Distance and Agents**” submitted in fulfillment of the requirements for the award of the degree of **Master of Technology in Electronics and Computer Engineering** with specialization in **Computer Science Engineering**, submitted in the **Department of Electronics and Computer Engineering, Indian Institute of Technology Roorkee, Roorkee**, is an authentic record of my own work carried out, under the guidance of **Dr.A.K.Sarje**, Professor, Department of Electronics and Computer Engineering, Indian Institute of Technology Roorkee, Roorkee.

I have not submitted the matter embodied in this project report any other degree.

Date: 26 - 06 - 08

Place: Roorkee

V. Aruna Kumari
(ARUNA KUMARI VAJJAGIRI)

CERTIFICATE

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Date: 26 - 06 - 08

Place: Roorkee


27/6/08
Prof. Dr.A.K.Sarje

Professor

Department of Electronics and Computer Engineering

IIT Roorkee – 247667

India

Acknowledgement

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crowned my efforts with success.

I take this opportunity to express my profound sense of indebtedness and sincerest gratitude to my guide, **Dr.A.K.Sarje**, Professor, Department of Electronics and Computer Engineering, Indian Institute of Technology, Roorkee for his expert guidance, valuable suggestions, inspirational support and constant encouragement throughout the work without which this work would not have been in its present shape. I deem my privilege to have carried out my dissertation under his able guidance.

I also wish to thank all my friends for their valuable suggestions, encouragement and timely help.

Finally, I would like to say that I am indebted to my parents for everything that they have given to me. I thank them for the sacrifices they made so that I could grow up in a learning environment. They have always stood by me in everything I have done, providing, constant support, encouragement and love.

ARUNA KUMARI VAJJAGIRI

Abstract

Searching for relevant information has become a skillful art in the current scenario of vast amount of available data. Often we may end up with a large amount of unrelated data. Many a times it requires a lot of human effort to choose from the presented information, after putting a lot of effort to find the information the user has no way to help other users who are searching for similar information. At most now a day's search engines (like Google) are giving results based on self history. In this we are proposing an Agent based frame work to make the users, find the relevant information more quickly not only by using his/her history but also by using other users search experience. For this we are using a measure called Query distance to find the similarity between the two queries. Problem occurs when the users of different domains submits the same query with different intensions for this we are using the categories. To make the Query distance more appropriate we are using the semantics of the query also when finding the query distance.

Table of Contents

Student declaration	i
Acknowledgement	ii
Abstract	iii
Table of Contents	iv
List of Figures	vii
List of Tables	ix
Abbreviations	x
Chapter1 Introduction	1
1.1 Introduction.....	1
1.2 Statement of the problem.....	2
1.3 Outline of the Dissertation.....	3
Chapter 2 Back Ground	4
2.1 Information Retrieval system.....	4
2.1.1 Web Based IR system.....	5
2.1.2 Agent based IR system.....	6
2.1.2.1 Issues and Architectures.....	6
2.2 Recall and Precision.....	7
2.3 JADE and the Agents paradigm.....	8
2.3:1 JADE Architecture.....	9

Chapter 3 Literature Survey.....	12
3.1 Related work.....	12
3.2 Motivation.....	15
3.3 Research Gaps.....	15
Chapter 4 Proposed System	16
4.1 System Architecture.....	18
4.1.1 Components of Proposed System.....	20
4.2 Query enhancing.....	20
4.3 Finding related queries and documents.....	22
4.4 Comparison of proposed system with existing system.....	22
4.5 Scope to implement in practice.....	23
4.6 Expected results.....	24
Chapter 5 Implementation Details.....	25
5.1 Implementing the Local Server.....	25
5.1.1 Community profile.....	25
5.1.2 GUI for Local server.....	27
5.2 Client implementation.....	27
5.3 Flow in the system.....	31
5.4 System Specifications.....	32
Chapter 6 Results and Discussions.....	33

Chapter 7 Conclusions and Future work.....	42
7.1 Conclusion.....	42
7.2 Future work.....	42
REFERENCES.....	43

List of Figures

Figure 2.1	Information Retrieval systems.....	3
Figure 2.2	Web information retrieval systems.....	4
Figure 2.3	IR parameters, Recall and Precision.....	7
Figure 2.4	Relationship between the main architectural elements.....	8
Figure 2.5	UML diagram showing the Relationship between the main architectural elements.....	9
Figure 3.1	Personalized IR with one level filtering.....	12
Figure 3.2	Agent based system with a common place to meet.....	13
Figure 4.1	Community profile.....	16
Figure 4.2	Proposed Agent based IR platform.....	17
Figure 4.3	IR system with two level filtering.....	18
Figure 4.4	Enhanced query.....	20
Figure 5.1	Snapshot of Community profile.....	25
Figure 5.2	Snapshot of Local Server.....	26
Figure 5.3	Snapshot of Google results.....	27
Figure 5.4	Snapshot of Community results	28
Figure 5.5	Snapshot of user viewing the related document.....	29
Figure 6.1	Snapshot of result helper.....	36
Figure 6.1	Graph for precision.....	37
Figure 6.2	Graph for recall.....	38
Figure 6.3	Snapshot of new user profile preferences.....	39

Figure 6.4 Recall graph after introducing new user.....40

Figure 6.5 Precision graph after introducing new user.....41

List of Tables

Table 6.1	User1 profile preferences.....	31
Table 6.2	User2 profile preferences.....	32
Table 6.3	User3 profile preferences.....	33

Abbreviations

IR	Information Retrieval
JADE	Java Agent Development Environment
HTML	Hyper Text Markup Language
XML	Extensible Markup Language
API	Application Programming Interface
DF	Directory Facilitator
AMS	Agent Management System
LADT	Local Agent Descriptor Table
GADT	Global Agent Descriptor Table
CT	Container Table
FIPA	Foundation for Intelligent Physical Agents
MTP	Message Transport Protocol
UML	Unified Modeling Language
PHP	Hypertext Preprocessor
ARB	Agent Resource Broker
SICS	Systems for Implicit Culture Support
ODP	Open Directory Project
GUI	Graphical User Interface

CHAPTER 1

INTRODUCTION

1.1 Introduction

The exponential growth of content in the web posed many challenges in finding the relevant information. Modern search engines are able to address this growth by effective indexing techniques. Google.Co indexes more than 6 billion items, including more than 4 billion web pages. Most of the modern search engines are keyword based. It means they return some set of documents for each keyword irrespective of the user who entered the query. As the Information on the web increases, more results will be found for the query. As the No. of documents increases for each keyword it is very difficult to find relevant information. So some help needed to user in finding the information. The basic aim of personalized search is to provide the assistance to user in finding information related to his/her needs.

Every user has a specific goal when searching for information in the web; the above statement lays the roots to the personalized web search. But as the No. of users in the web increases it is more likely that group of users may have similar interests. If similarities exist between user interests then it is more likely that one user search experience may help other user to find the information. Finding users having similar interests is difficult; users are not similar in all the cases. User interests may match in some instances and may not in some other. In order to find the related search sessions first it is important to find the context in which user is entered the query. The context can be found using the user past sessions.

Search engines like Yahoo and Google, projects like ODP(open directory project) trying to organizing web data in the form of Categories this will simplify the construction of user profiles to facilitate personalized web search.

The integration of agent technology and ontology could significantly affect the use of Web services and the ability to extend programs to perform tasks for users more efficiently and with less human intervention [4].

Recommender systems can also be considered as tools for the effective access to the available information. They can be classified as content-based, collaborative filtering, or hybrid systems. Content-based systems produce recommendations by analyzing the content of previously browsed pages and using the obtained information to find pages with similar content. Collaborative filtering systems calculate similarity between the different users and provide the user with the pages that have been selected by the similar users. Hybrid recommender systems exploit both approaches to a certain extent [1]. However, the majority of the recommender systems need user feedback and those systems that collect this feedback in explicit form force user to perform some extra work, like rating the items.

Most of the collaborative systems use agent technology to get suggestions from other users. Most of the agent based systems find related users in order to get the suggestions. As we mentioned earlier users are not similar in all the cases so in the proposed system we are not finding the related users to get the suggestions instead we are finding the related queries. By using user profile we will find the related queries, only for a set of categories that are more interesting to the user later we will find the queries of different categories.

1.2 Statement of the problem

As the information on the web increases the number of users using it also increases. The dynamic nature of the web poses many challenges in finding the required information. We propose a model system based on the agents and ontology to make the search sessions quick. We have taken a collaborative search approach in which the search session and optional feedback from one user may be useful to another user in similar context. The work towards the solution of this problem can be divided as follows

- Proposing an agent based environment, in which one user experience may help other users.
- Finding related queries rather than related users while getting suggestions.

1.3 Outline of the dissertation

This report is divided into seven chapters including this introductory chapter. The rest of this thesis is organized as follows.

Chapter2 explains about IR system, Agent structure, JADE (java agent development environment), Measures (Recall & Precision) this is necessary to understand rest of the material.

Chapter3 gives a brief description about related work, elaborates research gaps and motivation. This content lays the foundation to the work.

Chapter4 explains the proposed system, its architecture, basic blocks and comparison with existing system.

Chapter5 provides the Implementation details, working of the system and limitations of sample implementation.

Chapter6 presents the results of our work.

Chapter 7 concludes the dissertation and gives some suggestions for future work.

CHAPTER 2

BACKGROUND

This chapter provides the relevant background information to understand the proposed work.

2.1 Information Retrieval System:

Information Retrieval is the process of retrieving relevant information to the given query. The following shows the simple IR system.

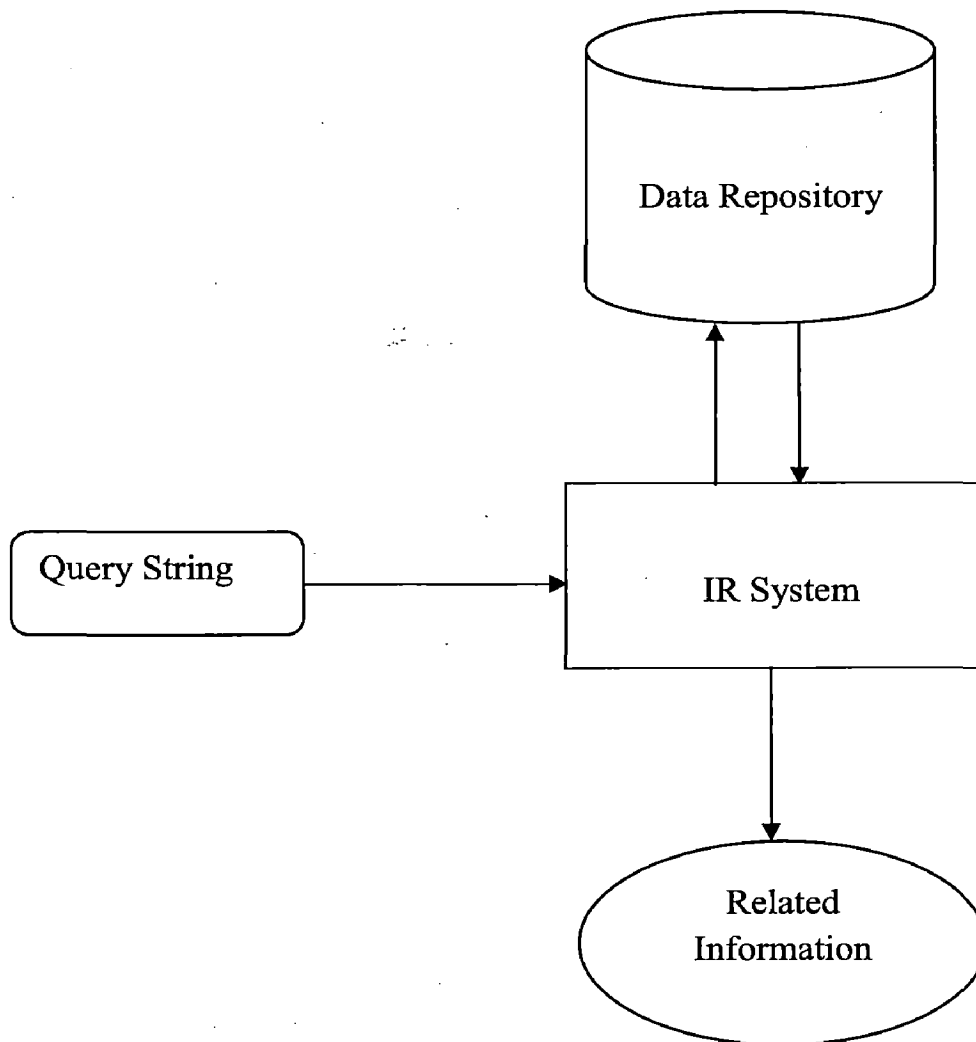


Figure 2.1 Information Retrieval System

Here Data Repository is a data base of data that is usually organized in some order. In practice this data base is very huge. So it is not possible to search for the relevant information manually. IR System is software written to search the data base for the relevant information. User normally enters a search query representing his interest the IR System uses this query to search the data base. After finding the relevant information it is presented to the user in some graphical form. The IR Systems optionally do the ordering of Retrieved data. For example the IR System for the Web ranks each retrieved document according to the user needs.

2.1.1 Web based IR System:

In web, HTML documents are assembled by spidering the web. The hyperlinks are considered as edges that connects the documents. We can exploit the structural layout information in HTML (XML). Can exploit the link structure of the web For example the Meta data contains information that often describes the document more precisely. The links it points to also help us determining the context of the document. But, it has the disadvantage of inconsistent data. Documents change

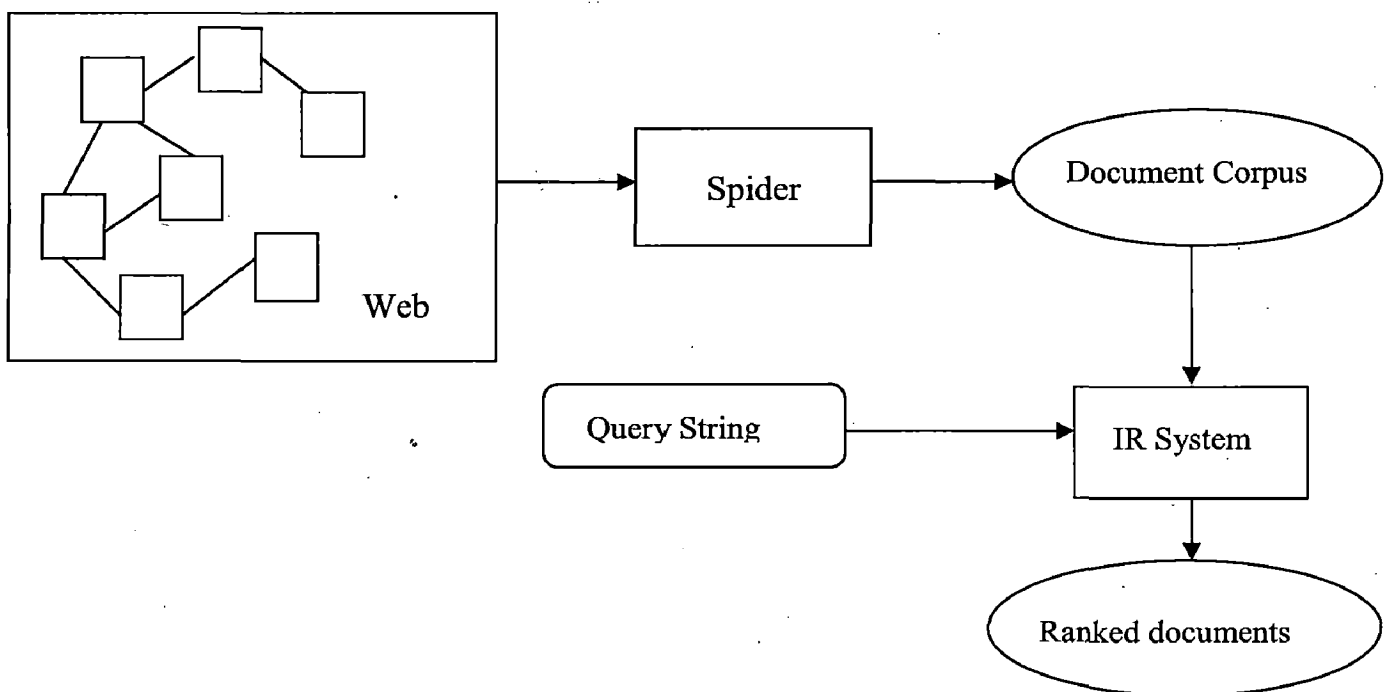


Figure 2.2 Web Information Retrieval System

uncontrollably. We may bring one web page and index it into our corpus. But later, when the user queries and retrieved a web document as a result, it may not even exist.

2.1.2 Agent based IR System:

Agents are semi-autonomous computer programs that intelligently assist the user with computer applications. Agents employ artificial intelligence techniques to assist users with daily computer tasks [6]. Such as reading electronic mail, maintaining a calendar, and filing information. Agents learn through example-based reasoning and are able to improve their performance over time.

Agents are a very promising technology for information retrieval. Some applications are intelligent IR interfaces, mediated searching and brokering, and clustering and categorization.

2.1.2.1 Issues and Architectures of agents

Two issues concerning agents are trust and competence [7]. Concerning trust [7] [8], the user and other members of the user community must be able to trust that the agent does only what the user wants done. The user must feel comfortable delegating tasks to the agent. Gaining trust is a crucial task in current scenario of widely distributed viruses and worms. As for competence, the agent must first acquire the skills to accomplish the delegated tasks [8] [9]. The agent may start with a generic behavior and may learn gradually mainly depending on the user feedback. The agent must also be able to decide when to help the user and how to help the user. There are three major paradigms for building agents [8].

The first approach is to make the agent an integrated part of the end-program. The advantage here is that the user trusts the agent because the rules are set. The problem is with competence. A combined agent and end-program requires too much insight from the user. The user must have the knowledge to effectively employ the agent. The second approach is a knowledge-based approach, where the agent has extensive domain-specific information about the application. Competence is a problem with this approach because it requires a huge amount of knowledge from the

knowledge engineer. Trust is also a problem since the agent is usually autonomous from the start, which gives users a feeling of loss of control and lack of understanding [10]. The final approach is a learning approach, where the agent has some knowledge of the domain but learns what the user would like it to do base on user actions. The learning approach has the advantages of the other two approaches while minimizing their disadvantages. The learning approach is the architectural paradigm that this method will use.

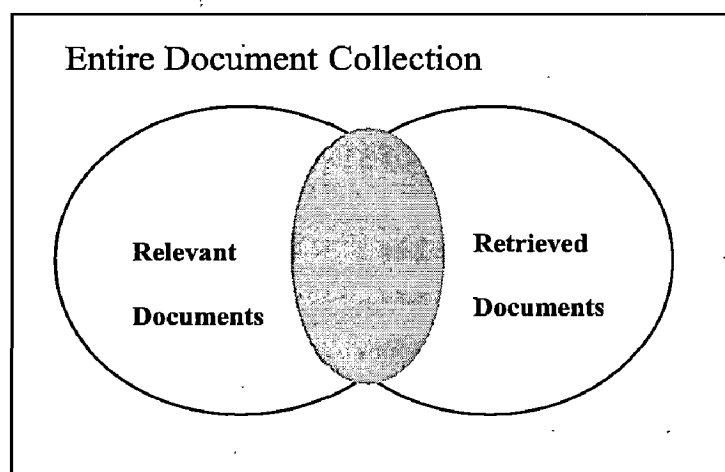
2.2 Recall & Precision:

There are two accepted standards of performance for comparing and evaluating retrieval systems, recall and precision [5]. Precision is the ability to retrieve top-ranked documents that are mostly relevant. Recall is the ability of the search to find all of the relevant items in the corpus. Hence, the definitions of recall and precision are,

$$\text{Recall} = \frac{\text{number of relevant documnets retrieved}}{\text{Total number of relevant documents}} \quad (2.1)$$

$$\text{Precision} = \frac{\text{number of relevant documnets retrieved}}{\text{Total number of documents retrieved}} \quad (2.2)$$

We can represent this graphically as follows.



Irrelevant	Irrelevant & Retrieved	Irrelevant& Not Retrieved
	Relevant& Retrieved	Relevant& Not Retrieved
Relevant	Retrieved	Not Retrieved

Figure 2.3 IR parameters, Recall and Precision

2.3 JADE and the Agents Paradigm:

JADE is a software platform that provides basic middleware-layer functionality which are independent of the specific application and which simplify the realization of distributed applications that exploit the software agent abstraction. A significant merit of JADE is that it implements this abstraction over a well-known object-oriented language, Java, providing a simple and friendly API. The following simple design choices were influenced by the agent abstraction.

An Agent is Autonomous and Proactive: An agent cannot provide call-backs or its own object reference to other agents in order to mitigate any chance of other entities cooperating control of its services. An agent must have its own thread of execution, using it to control its life cycle and decide autonomously when to perform which actions.

Agents Can Say 'No', and they are loosely coupled: Message-based asynchronous communication is the basic form of communication between agents in JADE; an agent wishing to communicate must send a message to an identified destination (or set of destinations). There is no temporal dependency between the sender and receivers: a

receiver might not be available when the sender issues the message. There is also no need to obtain the object reference of receiver agents but just name identities that the message transport system is able to resolve into proper transport addresses. It is even possible that a precise receiver identity be unknown to the sender, which instead may define a receiver set using an intentional grouping

Furthermore, this form of communication enables the receiver to select which messages to process and which to discard, as well as to define its own processing priority it also enables the sender to control its thread of execution and thus not be blocked until the receiver processes the message. Finally, it also provides an interesting advantage when implementing multi-cast communication as an atomic operation rather than as N consecutive method calls

2.3.1 JADE Architecture:

A JADE platform is composed of agent containers that can be distributed over the network. Agents live in containers which are the Java process that provides the JADE run-time and all the services needed for hosting and executing agents [15].

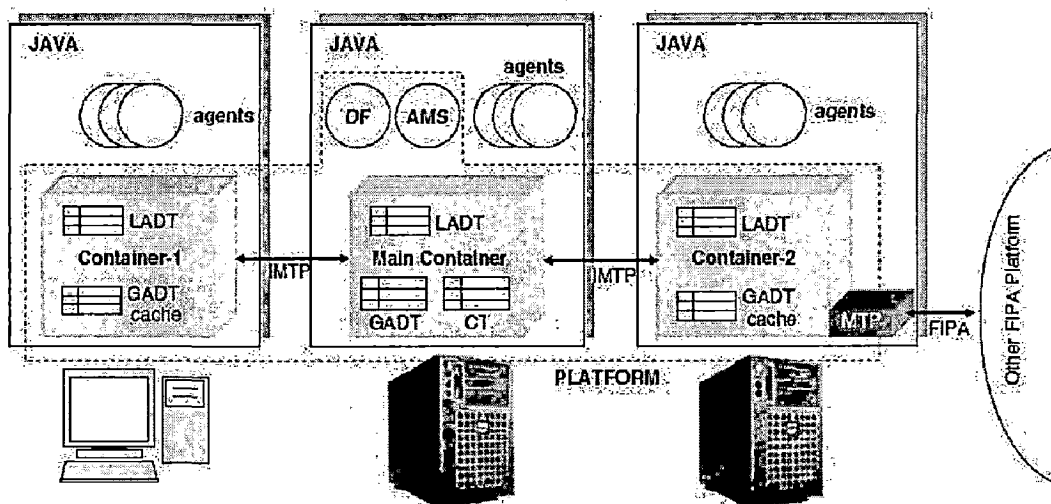


Figure 2.4 Relationship between the main architectural elements

There is a special container, called the main container, which represents the bootstrap point of a platform: it is the first container to be launched and all other containers must join to a main container by registering with it.

The programmer identifies containers by simply using a logical name; by default the main container is named 'Main Container' while the others are named 'Container-1', 'Container-2', etc. Command-line options are available to override default names.

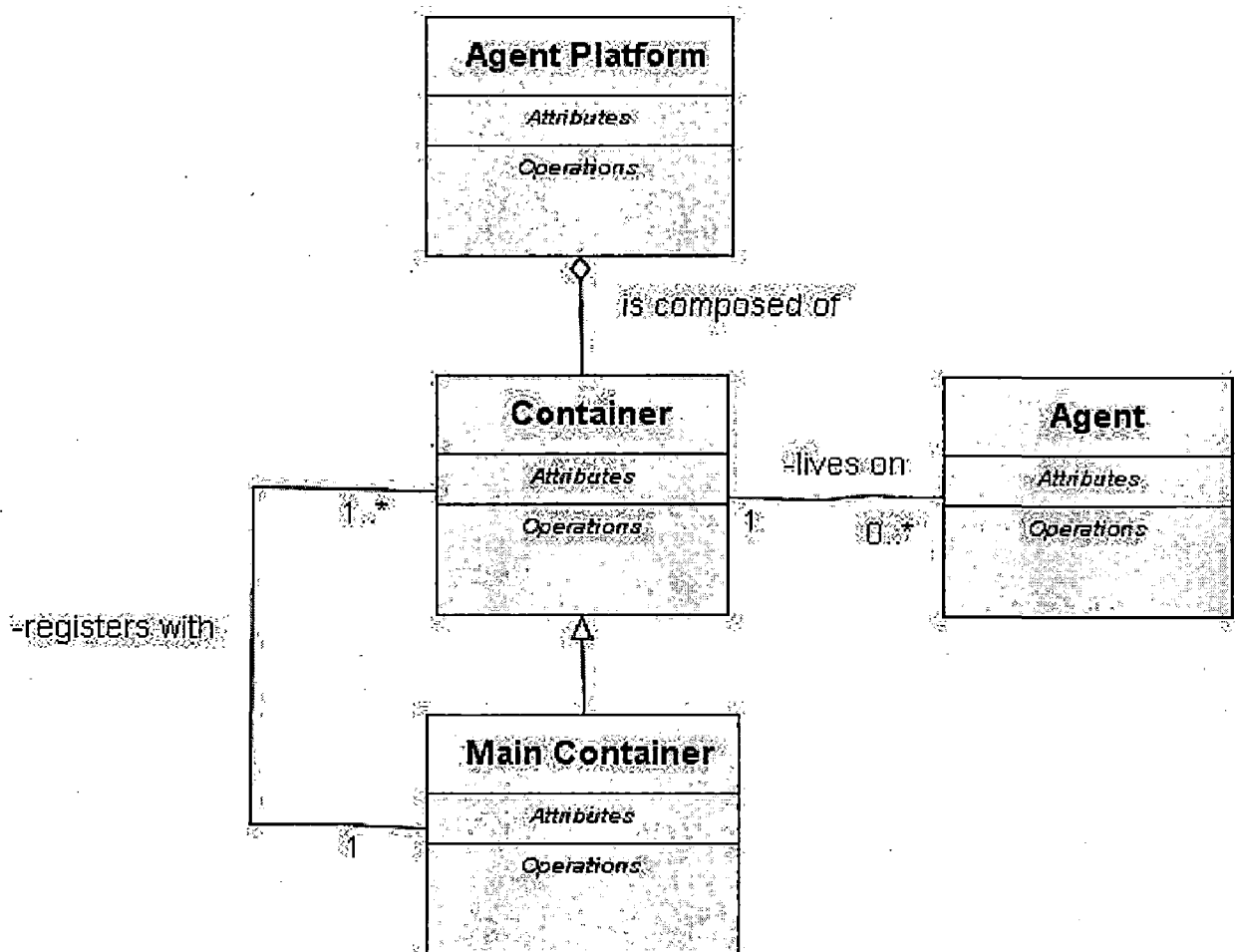


Figure 2.5 UML diagram showing the Relationship between the main architectural elements

As a bootstrap point, the main container has the following special responsibilities:

- Managing the container table (CT), which is the registry of the object references and transport addresses of all container nodes composing the platform;
- Managing the global agent descriptor table (GADT), which is the registry of all agents present in the platform, including their current status and location;
- Hosting the AMS and the DF, the two special agents that provide the agent management and white page service, and the default yellow page service of the platform, respectively.

CHAPTER 3

LITERATURE SURVEY

In this chapter, a brief overview of the related work done in this area is given and their demerits are also discussed.

3.1 Related Work

Most of the recent research in IR is focusing on personalized web search. Mainly there are two approaches for the personalized web search.

- Introducing ontology either to enhance the query or to reorder the returned results.
- Providing an agent platform in which agent can communicate with other agents and index servers to get the related information.

In both these cases user profile is constructed from user past experiences. The starting point of the profile may be a general profile or an empty profile [3]. In either case initially the results that are obtained are poor but after sufficient sessions the profile constructed sufficiently reflects the user's actual needs.

In the first case this profile is used to remove the ambiguity in keywords. Can enhance the query based on users most interested categories. So the search results are filtered using user profile. The below diagram shows this we can observe from the diagram that sending the query and receiving results are through user profile. While sending this profile is used to enhance the query like adding some keywords etc. while receiving the results this user profile can be used to reorder the returned results. So this can be seen as an Information retrieval system with one level filtering. Here the results re-ordering are done only based on only the user profile. So in this type of model the search sessions of one is can't help other user.

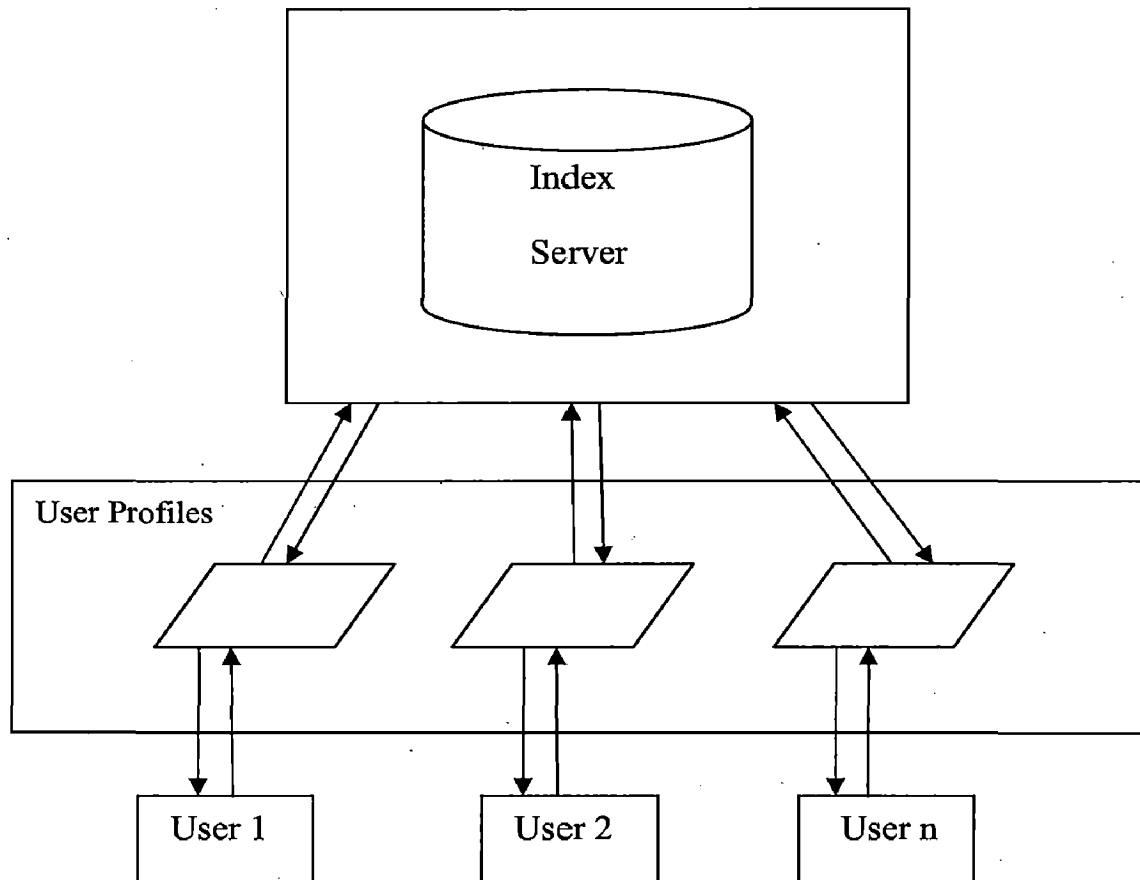


Figure 3.1 Personalized IR with one level filtering

In the second case an agent based IR system is used to improve the personalized search. In this agents learn about their user. These Agents talk to other agents and Index servers to get the appropriate results. The below architecture is proposed in [1].

From the below figure we can see that the architecture is provided a common meeting place to agents in which agents can talk to other agents to get the suggestions about a particular query. In [1] it is not given that how to find the related agents and how to weight the suggestions given by the agents. In practice this two issues are very important. Especially as the No. of agents increases it is even difficult to find related agents.

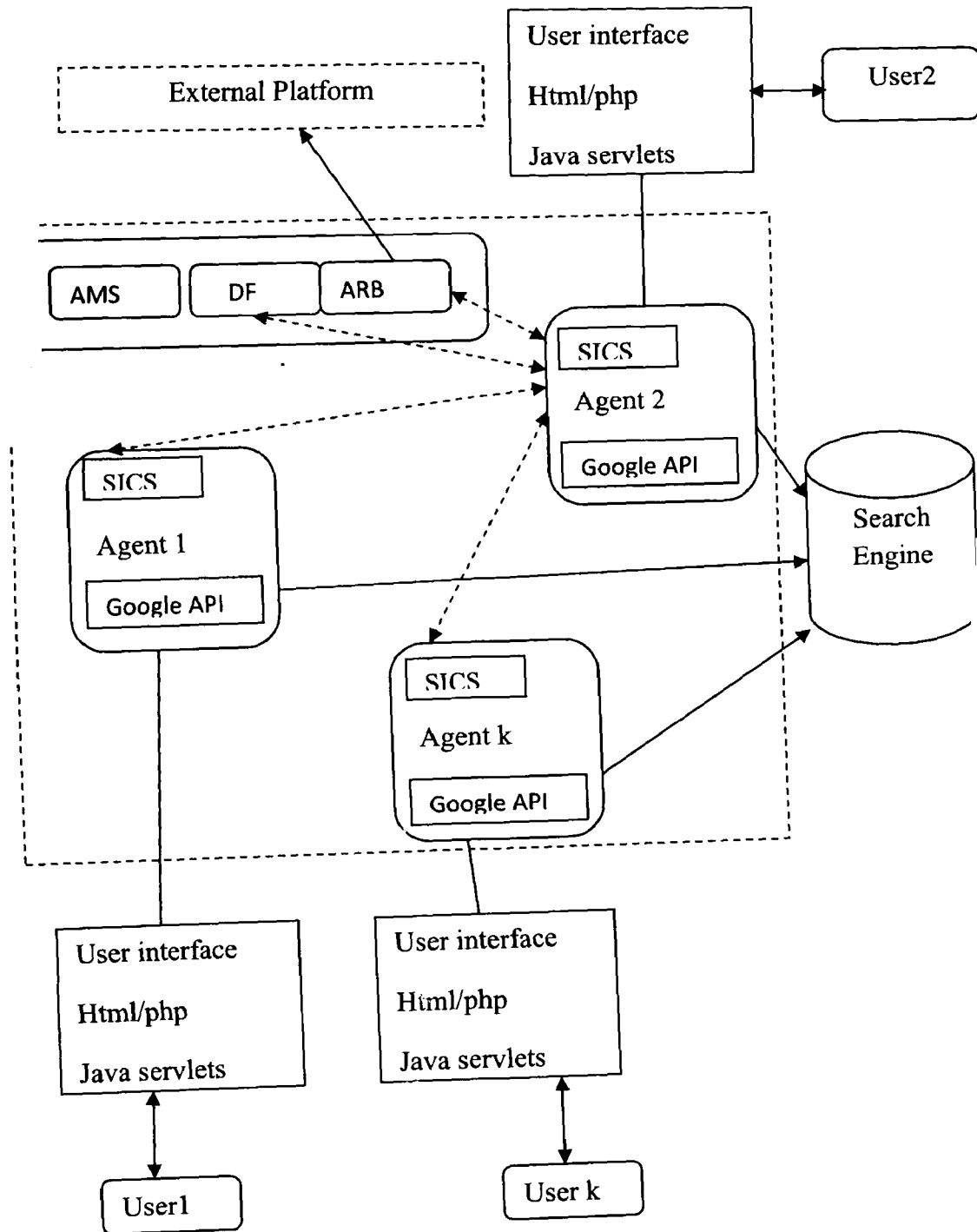


Figure 3.2 Agent based system with a common place to meet

3.2 MOTIVATION

There is so much research has been going on in this area to improve the personalized search, by introducing some ontology, to modify user query based on that ontology, using personal agents (either intelligent or multi agents) to learn about the user and improve the results. Much of this work is focused on the following main improvements.

1. Re- order the returned results (from search engine) based on user profile (/ontology) [2] [3].
2. Dynamically change user profile based on users past sessions [2].
3. Communicate between agents to collaboratively get results of similar users. This communication is mainly between agents of different users [1].
4. Providing a meeting place to agents in which mobile agents can come and get required information [1].

3.3 Research gaps

1. **When to reorder:** The systems which proposed to reorder the results are missing some important point. This can only be seen when talking about Semantic web. Let us consider a user entering query “Apple”. The systems which proposed the algorithms to reorder the results mainly work first by understanding the user and finding the related areas. And re- orders the results according to these areas. Usually search engines returns results for a query in 100’s or 1000’s. Most of the results are not belongs to the user interest. Much of the computing power is wasted in re-ordering this unnecessary document links. It is much easier and economic to eliminate these results in first place (in the server itself) according to the user interests. To do this server needs to identify each user separately it is worthy if it is done in a stateless fashion.
2. **Identification and Synchronization:** The agent based systems, like the systems in [19] says that user agents communicate with other user agents to get the related information. Not all agents can provide related information because

different user needs will be different so agents can provide information about their interested areas only. So it is clear that first the user agent needs to select appropriate agents among the other agents. This task is very difficult as the no. of agents increase. Even it is not clear that how to found weather this agent is appropriate per a particular query if user needs are varies among the queries. Even if we able to found the appropriate agents the agents need not be there when we want to search for the information so how to synchronize the users to search for the information at a particular time only this is not possible.

Chapter 4

PROPOSED SYSTEM

In this chapter, the proposed system details are specified, the proposed system consists of three agents. Two agents are present at client side and a common agent at server side. These agents are not mobile agents. The client side agents construct user profile from user past sessions. These past sessions may be past search sessions, user documents etc. This user profile is later used to enhance the query. The agent on the server side built the community profile which consists of the records of the following form.

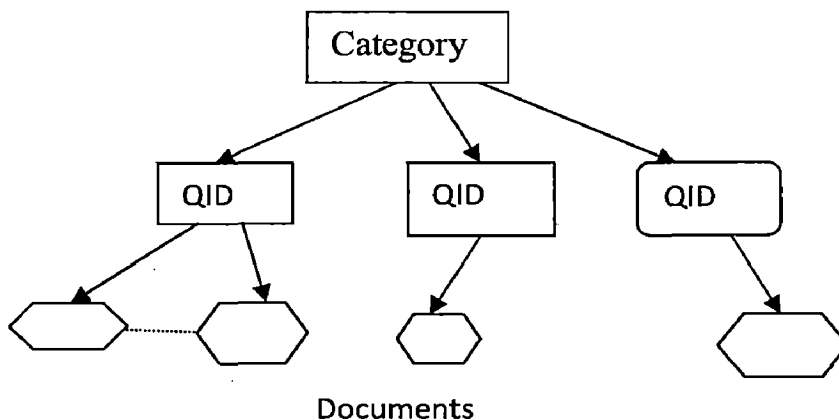


Figure: 4.1 Community profile

Each document is assigned a weight. This weight is assigned based on the user(s) who uploaded it, how much command the user(s) has in that category. This type of records is maintained for each category. From the user profile appropriate categories for each user can be found. After finding this category we query the server for results from these categories. This approach will help the new users to get the results more quickly by getting the help from more experienced users in the community and also experienced user get help from other experienced users. And by this type of approach only the index server needed to be up all the time not other user agents.

4.1 System Architecture:

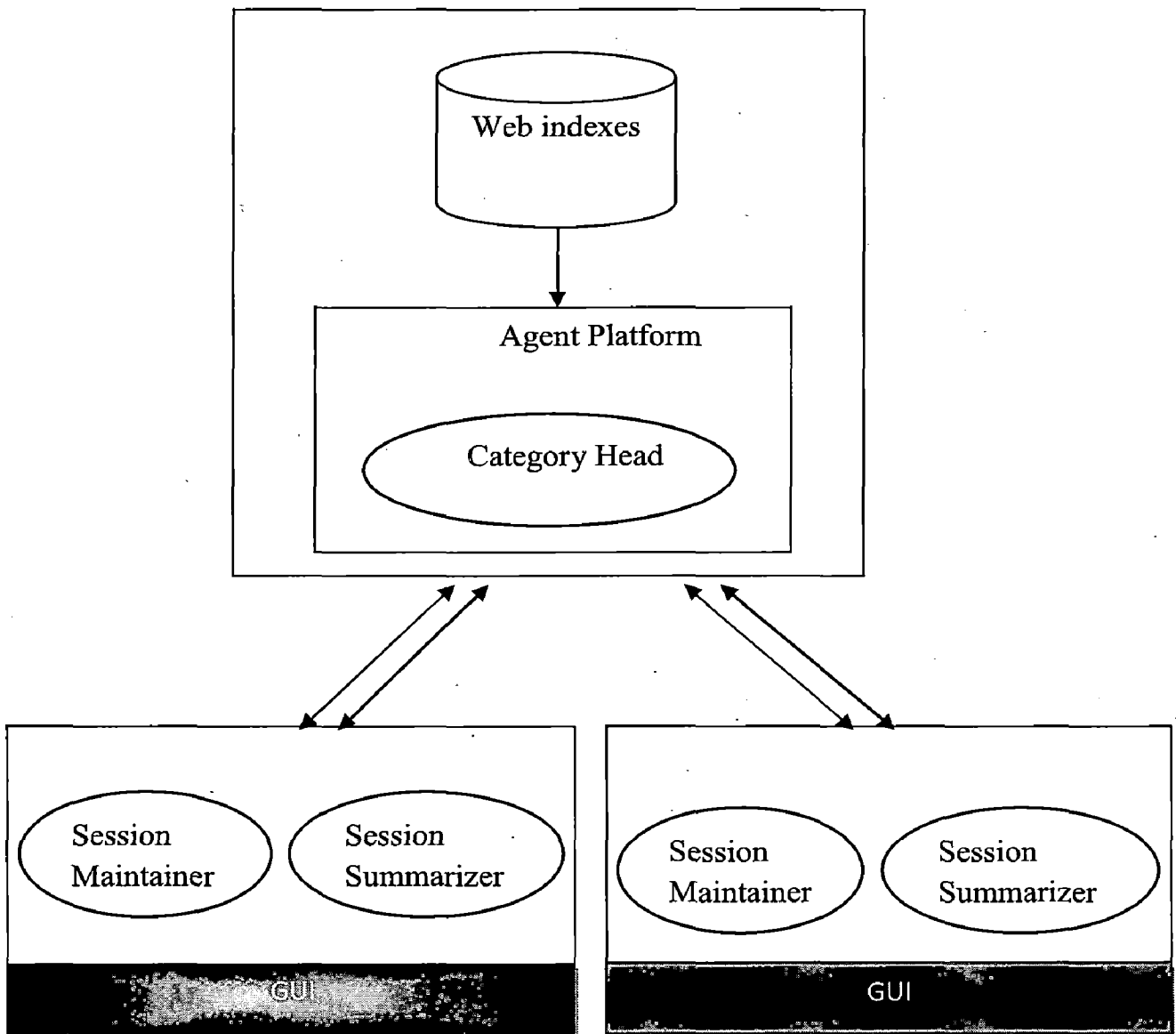


Figure 4.2 Proposed Agent based IR platform

By the above diagram we can see that there are three agents to help a user to retrieve the information. Two client side agents create user profile and enhance the query at user place.

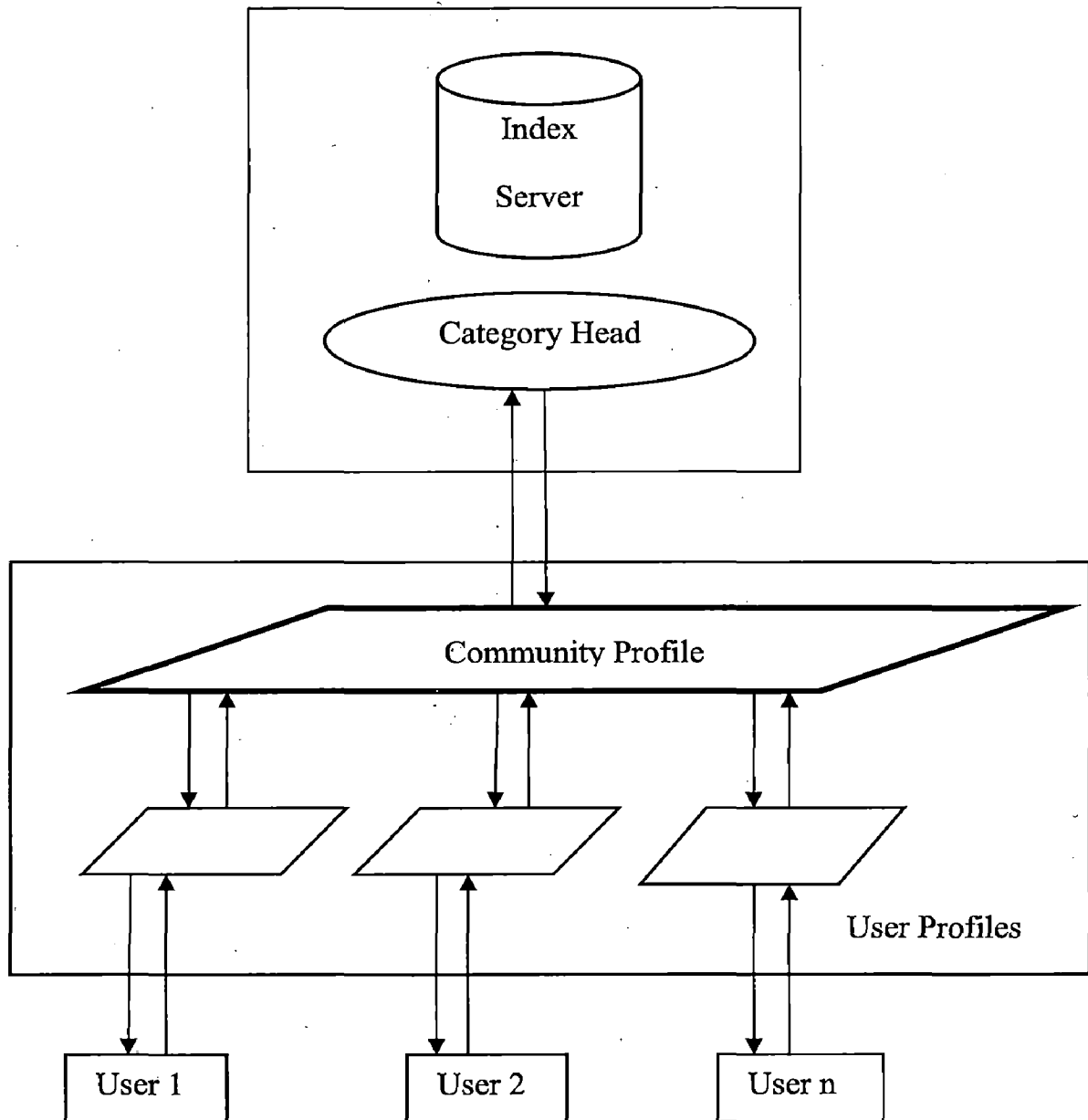


Figure 4.3 IR system with two level filtering

The other agent on the server side creates the community profile. By seeing the query he can find the related documents to the query, so no need to store information regarding user. This Community profile will form other layer to filter, so as shown in the above figure, this proposed system is IR system with two level filtering.

4.1.1 Components of proposed system

User Profile: it includes the different categories and their weights. These weights are dynamically updated based on users search sessions and summarized user past search sessions.

Community Profile: It is the profile constructs and maintained by the CategoryHead. It includes the Query Entries from different users.

Session Maintainer Agent: This agent resides on the client machine and it is responsible for preprocessing user queries, displaying the results to the user and constructing QueryEntry.

Session Summarizer Agent: This agent also resides on the client machine and it is responsible for calculating the weight of the QueryEntry and uploading the QueryEntry to server.

CategoryHead Agent: This agent resides at the server and is responsible for generating Community results based on Query Distance.

4.2 Query enhancing:

As we mentioned earlier user profile is used to enhance the query. The main requirement here is by seeing the query the server need to identify the related documents which are particularly interesting to the user and also it as to remove the ambiguity in the keyword based search.

In order to achieve the goals it is clear that it is not sufficient to send simple string to the server. So we constructed a query record.

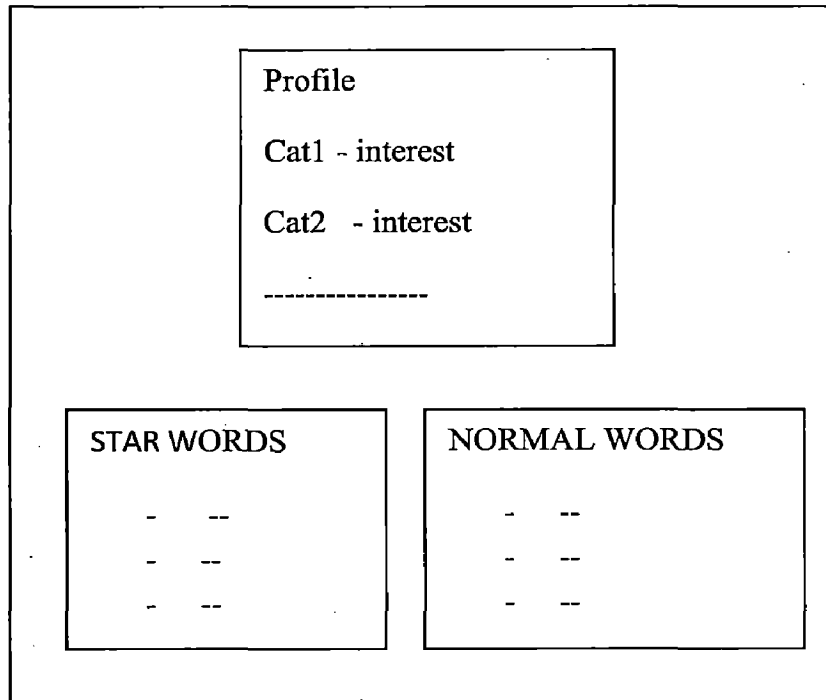


Figure 4.4 Enhanced query

From the above query the server can find out what are interests of the user. As we can see there is no explicit reordering is required the profile is sent in the order. So server can simply find out the results of each category and sent them. We are having the document weight for each document with in the category results this weight can be used to reorder.

Here we have taken two types of query words “STAR WORDS” are the words that must be match when finding the related queries. “NORM WORDS” are the words which may have different forms in English means same meaning can be expressed with different words in English this type of words are taken as normal words. In order to remove that ambiguity between the words which give same meaning we have introduced a general English ontology “Wordnet” to add all the words which give same meaning.

4.3 Finding Related Queries and Documents:

Community query is used to identify related queries; we know that different users form different types of queries to get same Information. So if we only concentrated on queries which are same then we are missing so much information. In order to reduce this loss we have considered a different approach after knowing the user interests from the Query. Server finds out the related broad categories with in those categories to found required document links it uses a metric called “QUERY DISTANCE”.

Query Distance: It is the similarity between the queries and it is used in finding the Community results. Here Jacquard Coefficient is used to find the similarity.

$$Sim = \frac{P}{Q + R + P}$$

P: number of terms common in both queries

Q: number of terms that are present in the query1 and not present in the query2.

R: number of terms that are not present in the Query1 and present in the query2.

4.4 Discussion on Proposed system and comparison with existing systems

- We are proposing to place the Community profile in the Index server itself. So whenever a query is received at the search server it can use this Community profile in the first place itself to filter the results. It will decrease the overall waiting time because in general the server is much more powerful than client machines. So it is more likely to place in the server.

more powerful than client machines. So it is more likely to place in the server.

- The system proposed in [1] also used somewhat similar approach but the main difference is it is a pure agent based proposal in which agents communicate with other agents to get the related results. The main problem here is we need to exactly found out the other agents that are similar and it is not particularly easy task in practice especially when the system grows exponentially. This system suffers from lack of scalability. In our proposed system we placed all the suggestions in a central place that will free the user agents from finding the appropriate agents. It will not suffer from single point of failure because if it suffers then the index server also suffers but already there are lot of search sites are running effectively like Google, yahoo. The same replication can be used here also.
- The system proposed in [3] is somewhat similar in constructing user profile. But it is very much different in usage of this profile. In [1] the user profile is used to re order the returned results. In this proposed to system this profile is used to construct the query based on user interests and server itself serves the query accordingly.
- There is no need for server to maintain the state information about any particular user so the proposed system is not too much different from present system in implementation but it gives results well.

4.5 Scope to implement in practice

- Community profile can be seen as another strategy to rank the documents.
- It is not required to maintain huge data base for the community query as when the suggestion came, it can be analyzed and the appropriate document weight can be altered for broad category of users.
- The user side agent environment can be implemented as a browser plug-in.

4.6 Expected Results

As the users in the community increase, it may be possible to have users with somewhat similar interests. As the similarities increase the quality of results will increase. It is not required to have exactly similar users combination of users with varying interests are also some times increase the results. For example user1 is interested in Games and computers as first interests and Arts as second preference and business as third. User2 has interests in the reverse order. Then while user1 is searching for information related to business he will get help from user2. In the same way if user2 is searching for information related to games he will find suggestions from user1. In the same way there are lot of permutations are possible.

Chapter 5

IMPLEMENTATION DETAILS

In this chapter, the implementation of the project will be provided and the working flow of the system is explained.

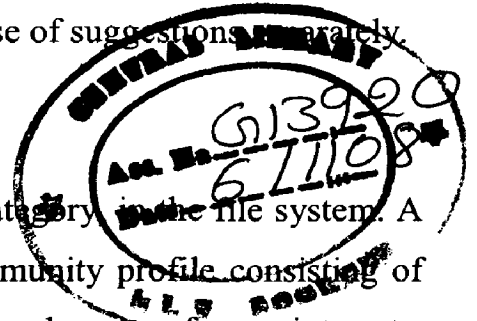
5.1 Implementing the local Server

It is quite difficult to index the whole web for this dissertation so we used the Google Index server and constructed a local server separately to maintain Community Profile. For this and client side agents, we used JADE platform it is a java based agent development platform. It is implemented as for FIPA standards.

Because we are separating the index server and community profile in our sample implementation the client agents need to send queries to index server and community profile differently and the user selections also needed to upload to the local server and the local server has to maintain a data base of suggestions separately.

5.1.1 Community profile:

Community profile is stored according to the category in the file system. A snap shot of the community profile is below. The community profile consisting of different categories these categories can be seen as broad areas of user interests. Within each category we have queries from different users. These queries represent the summaries of different users search sessions at one place. Again these queries consist of document suggestions from different users. That means within a single query it is possible to have document suggestions from different users. If two users are suggested a same query then the user who has highest experience in the category that user category weight will be given to the document. This document weights can be used to reorder the results within the category.



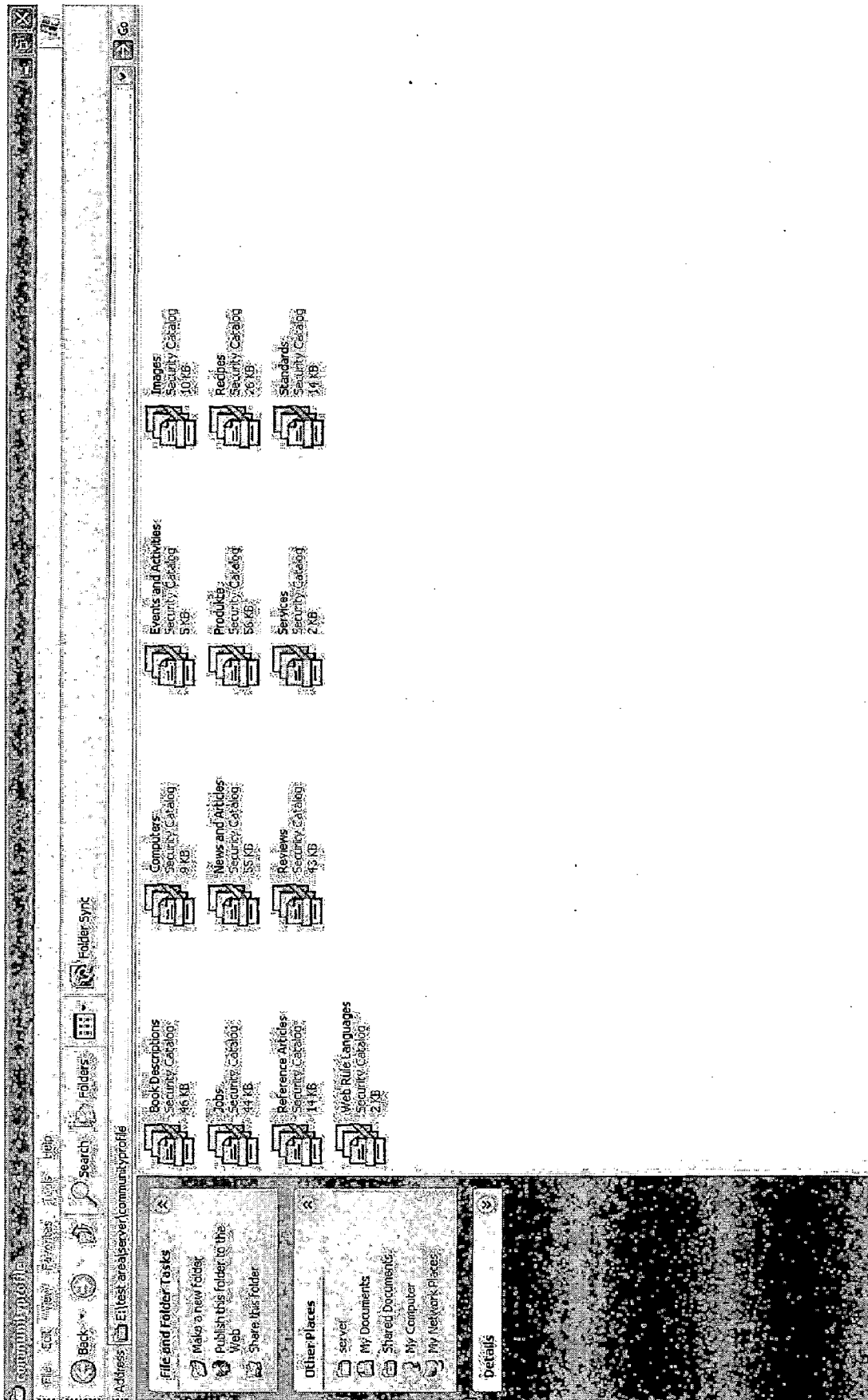
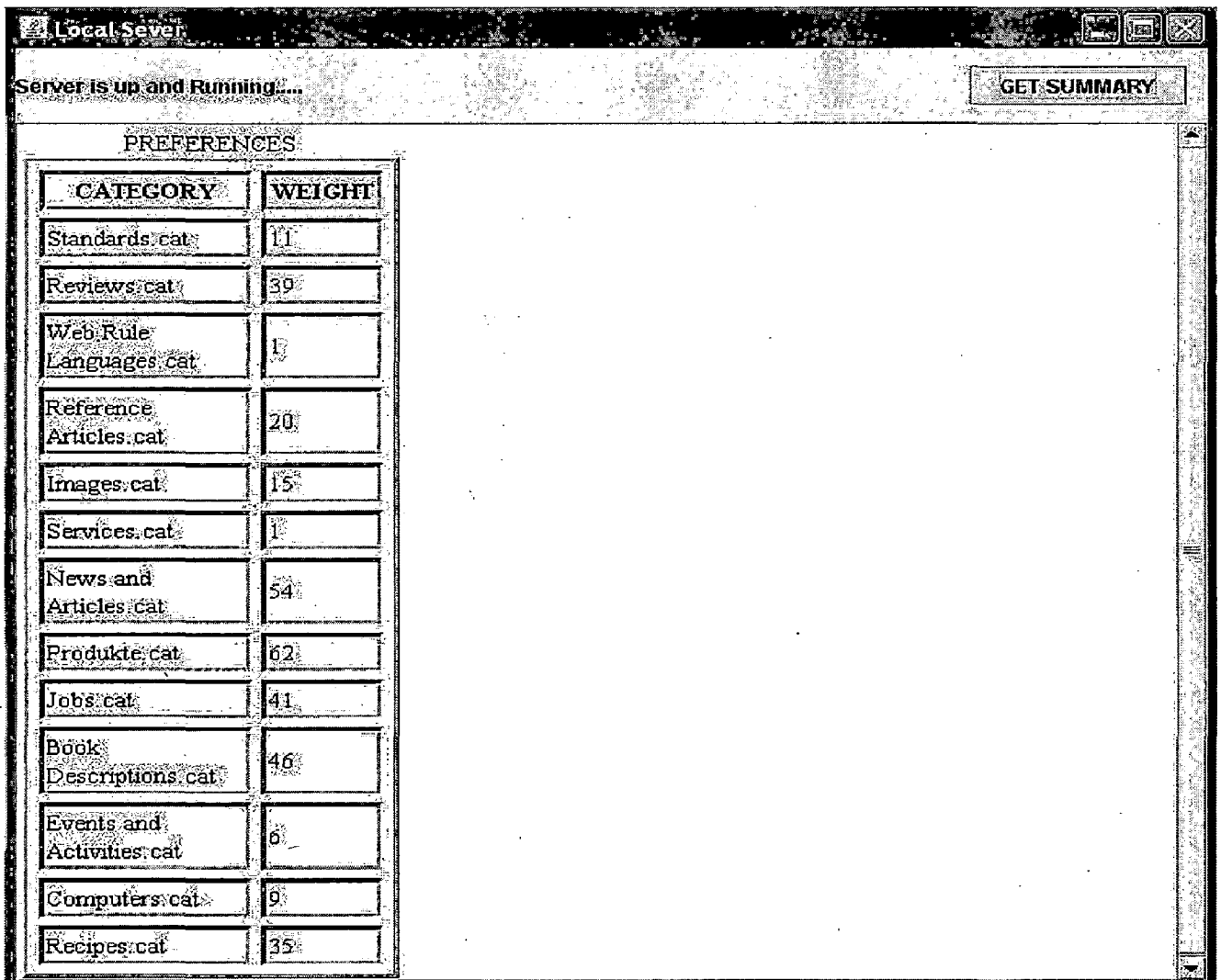


Figure 5.1 Snapshot of community profile

5.1.2 GUI for Localserver:

A simple GUI is implemented to show the status of the community profile. It shows the count of records in each category. This GUI shows No. of document suggestions in each category. It is not required in practice. A simple log file will be sufficient. We also had written sessions to log files in order to make debugging easier.



CATEGORY	WEIGHT
Standards:cat	11
Reviews:cat	39
Web Rule Languages:cat	1
Reference Articles:cat	20
Images:cat	15
Services:cat	1
News and Articles:cat	54
Produkte:cat	62
Jobs:cat	41
Book Descriptions:cat	46
Events and Activities:cat	6
Computers:cat	9
Recipes:cat	35

Figure 5.2 Snapshot of Local server

5.2 Client Implementation

In practice it is feasible to implement a plug-in for the browser. But in this sample implementation we implemented a simple web browser to show the results. It

has three tabs, Tab1 shows Google search results tab2 shows community results, tab3 shows the web pages.



Figure: 5.3 Snapshot of Google results

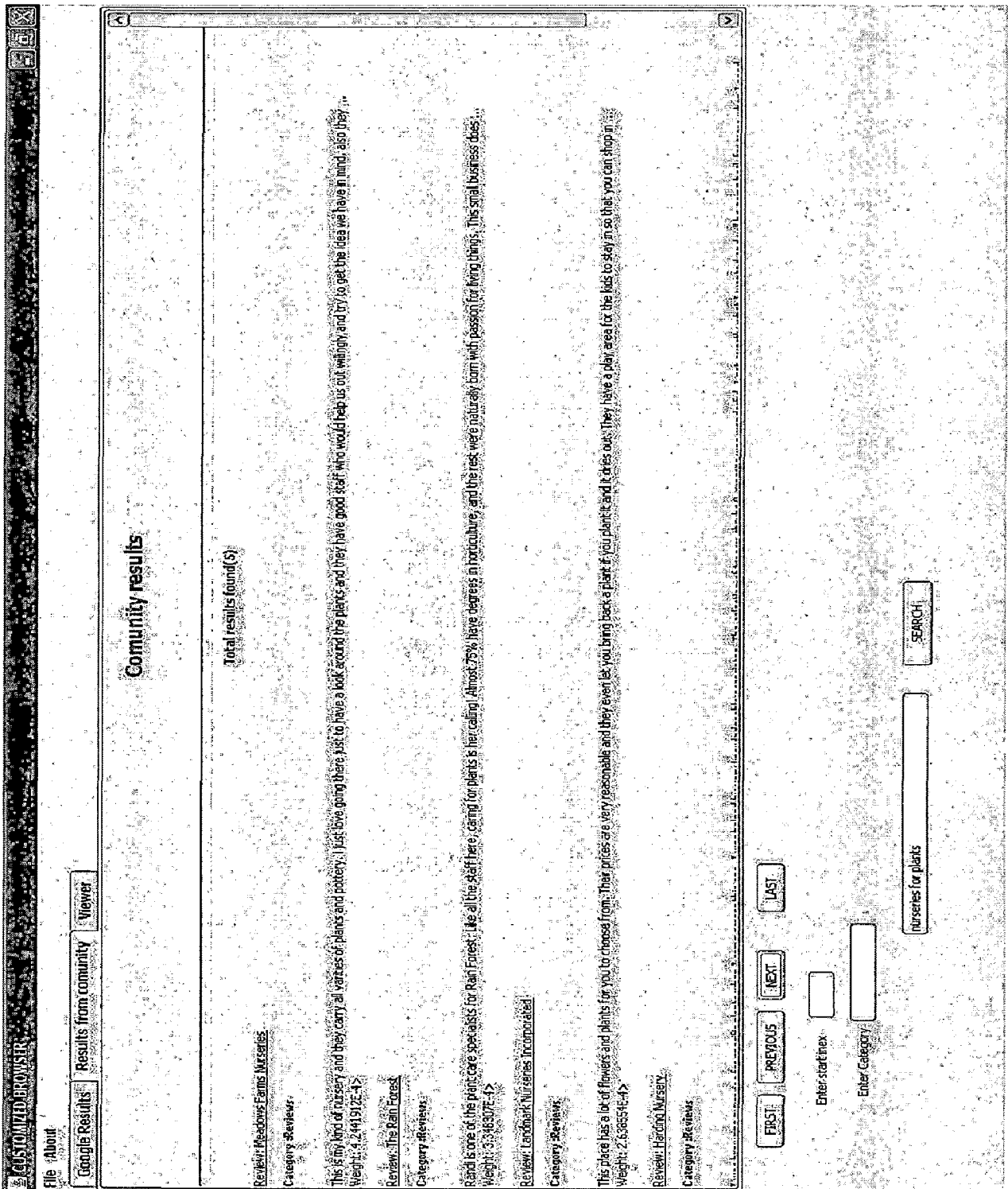


Figure 5.4 Snapshot of Community results

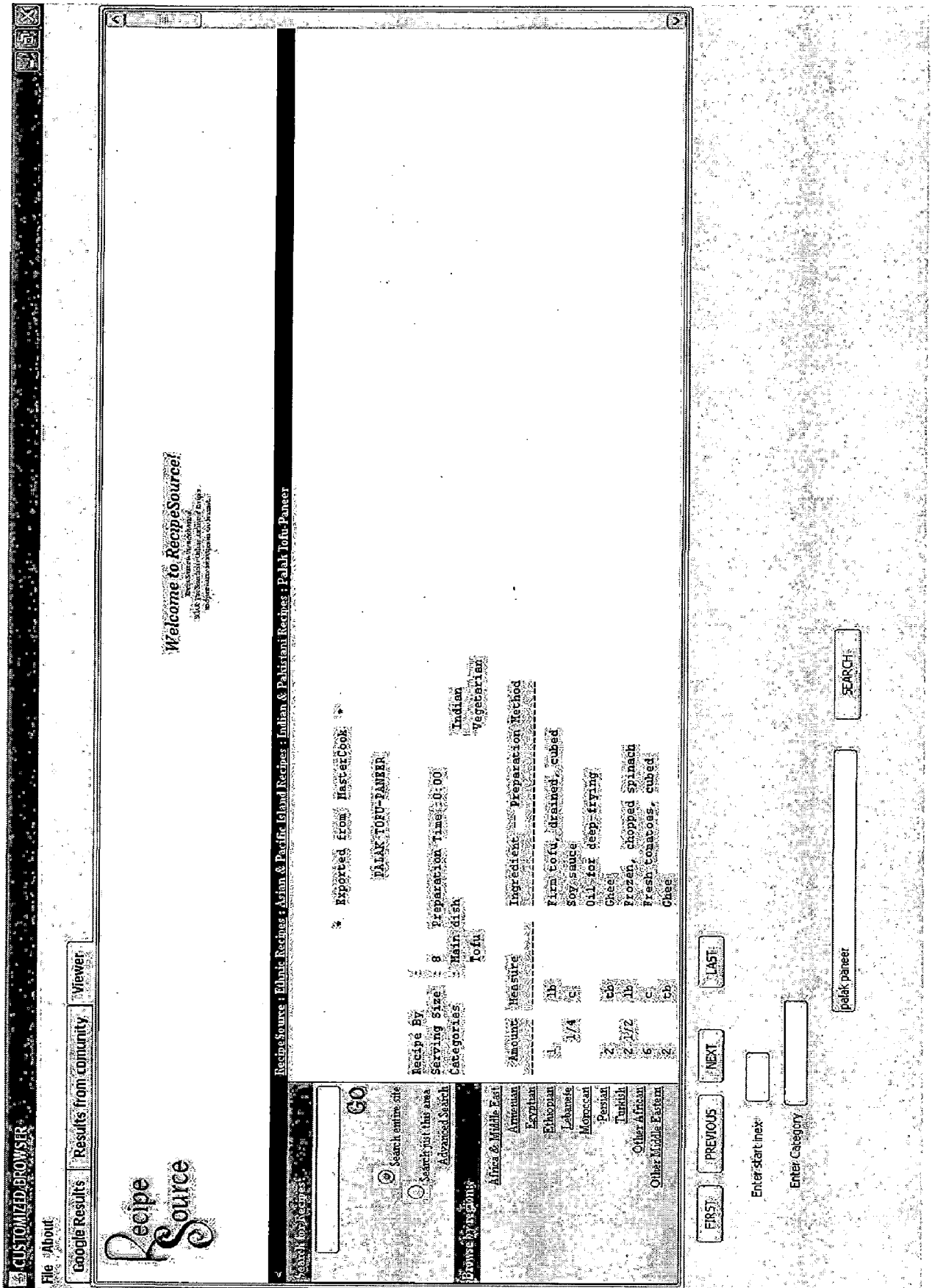


Figure 5.5 Snapshot of user viewing the related document

The figure 5.3 shows the Google index server results for the query entered by the user. We have also provided a text box to enter the category explicitly. This will help to form the category query to the Google Index server. In order to show the difference between Community results and Google results we are not reordered any results. We found that with out explicit reordering the community results are more close to the user needs. The diagram 5.4 shows the community results. We can observe that the No. of results from community are less in this case it is because we are seperated Index server and Index server in this implementation that's why theLocal server can only give the links which are seen by this user or other related user. This limitation will not be there we combine these two. The diagram 5.5 shows the snap shot of user viewing the related document.

In the next section we will explain the exact flow in the system beginning with the user query to the results.

5.3 Flow in the system

1. User enters the query in the GUI.
2. The "Session Maintainer" takes the user query and preprocess it
 - a. Removes stop words
 - b. Uses the Wordnet ontology and enhances the query
 - c. Constructs the request object using above query and category profile
3. Session Maintainer sends the object to server
4. Server returns two types of results

Type1 results:

- These are the general results with ordinary ordering (these results are equal to the results we obtain while searching in the Google like search engine)

Type2 results:

- These are the results from the Community profile.

5. Session Maintainer displays these results to user.

6. User selects one or more documents.
7. After user selecting a document Session Maintainer constructs a QueryEntry.
8. This Query Entry is sent to “Session Summarizer” agent
9. Session Summarizer agent calculates the weight of that document and forwards this document to “CategoryHead”.
10. Category Head inserts this record under appropriate category.

From the flow it is clear that in our proposed system actually consists of two level filtering. The first filtering is through user profile and the second level filtering is through the community profile. This implementation also implemented those two levels of filtering in this sample IR system.

5.4 System specifications

The following are the minimal system requirements.

Supported Operating systems: Windows NT/XP/VISTA, Linux

Hardware Requirements:

1. 500MHz Pentium class CPU or better
2. 512MB RAM
3. 2GB free disk space for the minimum JADE environment

In order to run Local server it is recommended to have higher RAM and CPU.

Software Requirements:

1. gdata 1.16.4 (Google API)
2. JADE- all 3.6
3. Wordnet 2.1
4. JWNL (for accessing Wordnet)
5. Java Mail API (Used by JADE)

Chapter 6

Results and Discussions

In order to test the sample implementation we created three users with the following profiles.

USER1

CATEGORY	WEIGHT	CLICKS
News and articles	2.676012E-19	14
Jobs	3.174385E-19	41
Reviews	2.8742414E-19	22
Computers	1.7265767E-19	4
Book descriptions	2.297868E-19	7
Images	2.2049743E-19	6
Recipes	2.800532E-19	18
Products	3.2822744E-19	53
Reference articles	2.4417482E-19	9

Table 6.1 User1 Profile preferences

USER2

CATEGORY	WEIGHT	CLICKS
Products	5.249354E-16	2
Events and articles	7.4237075E-16	4
News and articles	1.2470787E-15	50
Jobs	8.7953784E-16	7
Computers	5.249354E-16	2
Reviews	0.0	1
Book descriptions	1.275808e-15	60

Images	1.02427713E-15	16
--------	----------------	----

Table 6.2 User2 Profile preferences**USER3**

CATEGORY	WEIGHT	CLICKS
Jobs	0.0	1
Reviews	2.5173494E-14	1
Computers	1.877214E-14	9
Standards	2.5173494E-14	52
Recipes	2.2265327E-14	22
Products	2.1316516E-14	17
Reference Articles	1.4910916E-14	4

Table 6.3 User3 Profile preferences

The profile represents the broad categories the user interested in. clicks represents the No .of documents the user selected in each category and weight gives the command the user got in the category. After creating the user profiles we have selected 10 queries randomly from the queries we have given to build the profiles. Search sessions with these queries yielded following graphs for recall and precision.

Relevant links are calculated as follows.

- Using the user profile we found the categories to which this query may fall.
- After finding the categories we have calculated no of documents he/she may select from each category.
- After finding the no of relevant documents we can calculate precision.

Example:**For User1:****Query: world cup in +2006**

From user profile we found the following relevant no of links by category wise

Category	No
Products	2
News and Articles	2
Jobs	1
Images	1
Book Descriptions	1

For this query from the community 15 results found.

Now the precision is calculated as $7/15 = 0.4666$

To calculate Recall we have taken a different approach; the recall is the ratio of relevant links suggested to the total no of relevant links. We can't know the total No. of relevant results, so we have added a constant value to the relevant links suggested from user profile.

Now for the above example the recall = $7/7+20 = 0.2121$

Here 20 is the constant we have added. Here the No's may look small, it is because the community profile is containing small no of links. It will change if we merge the local server and index server. As the no of links in the community profile increase the constant that we are adding to the recall is increases.

To assist while calculating results we have created a small program which gives the no of relevant links for a particular user by analyzing the user profile for a particular query. The snap shots of this simple program are shown below.

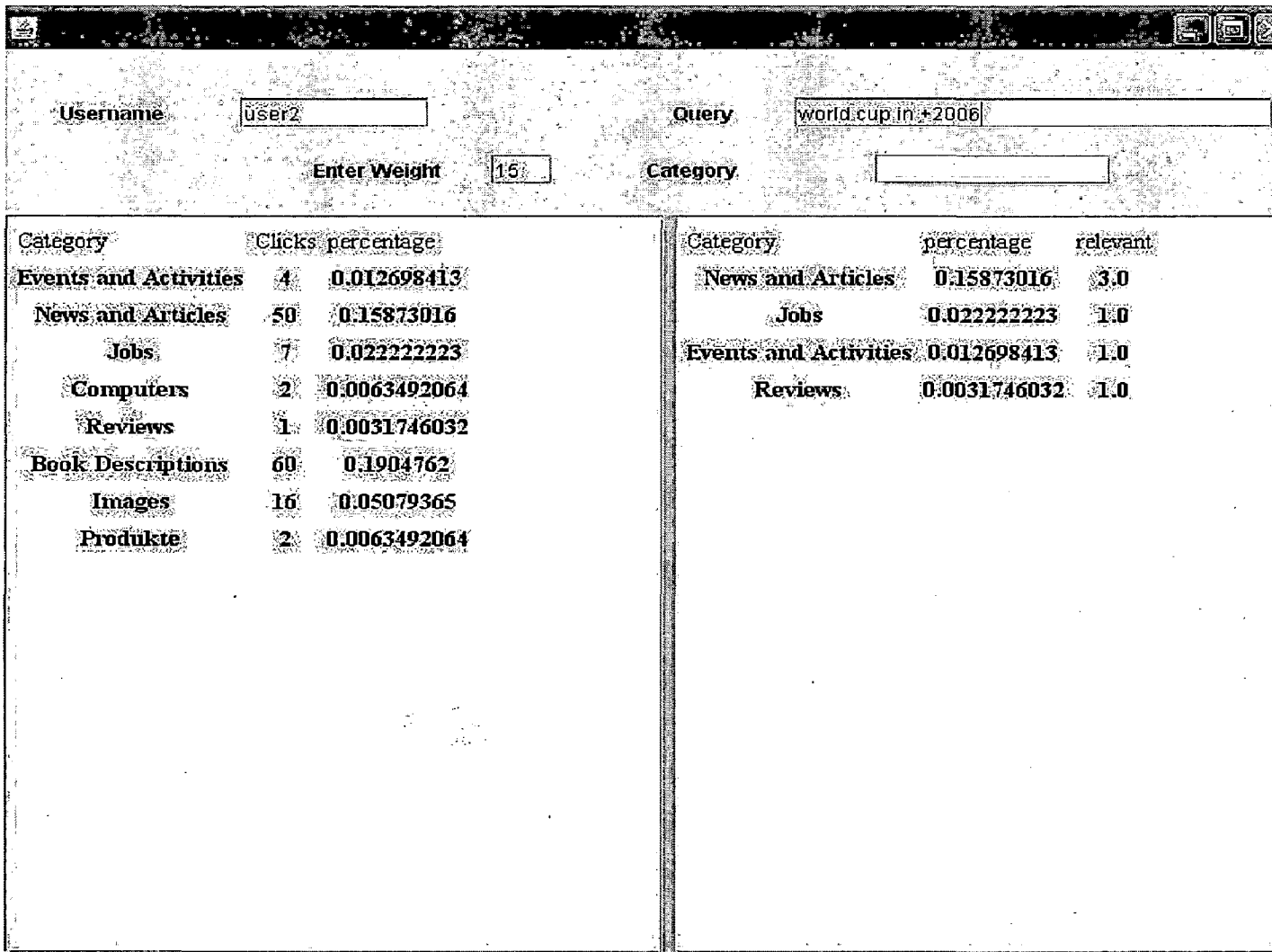


Figure 6.1 Snapshot of result helper

The left hand side shows the user2 profile on the right hand side shows the No of relevant links from each category.

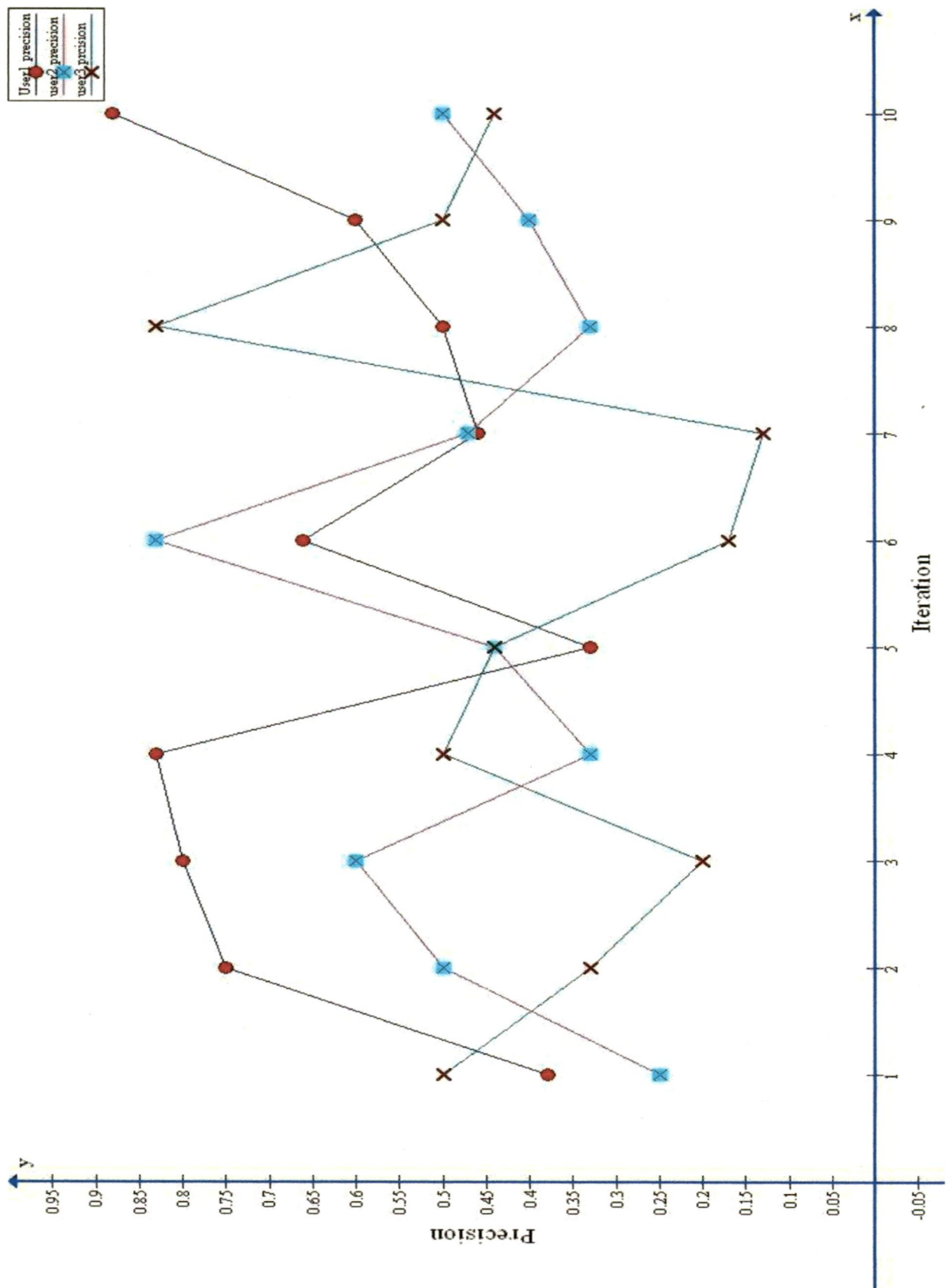


Figure 6.2 Graph for Precision

For each 10 queries we have calculated the precision value as follows we know that precision is the ratio of the no of relevant links suggested to the total no of links suggested.

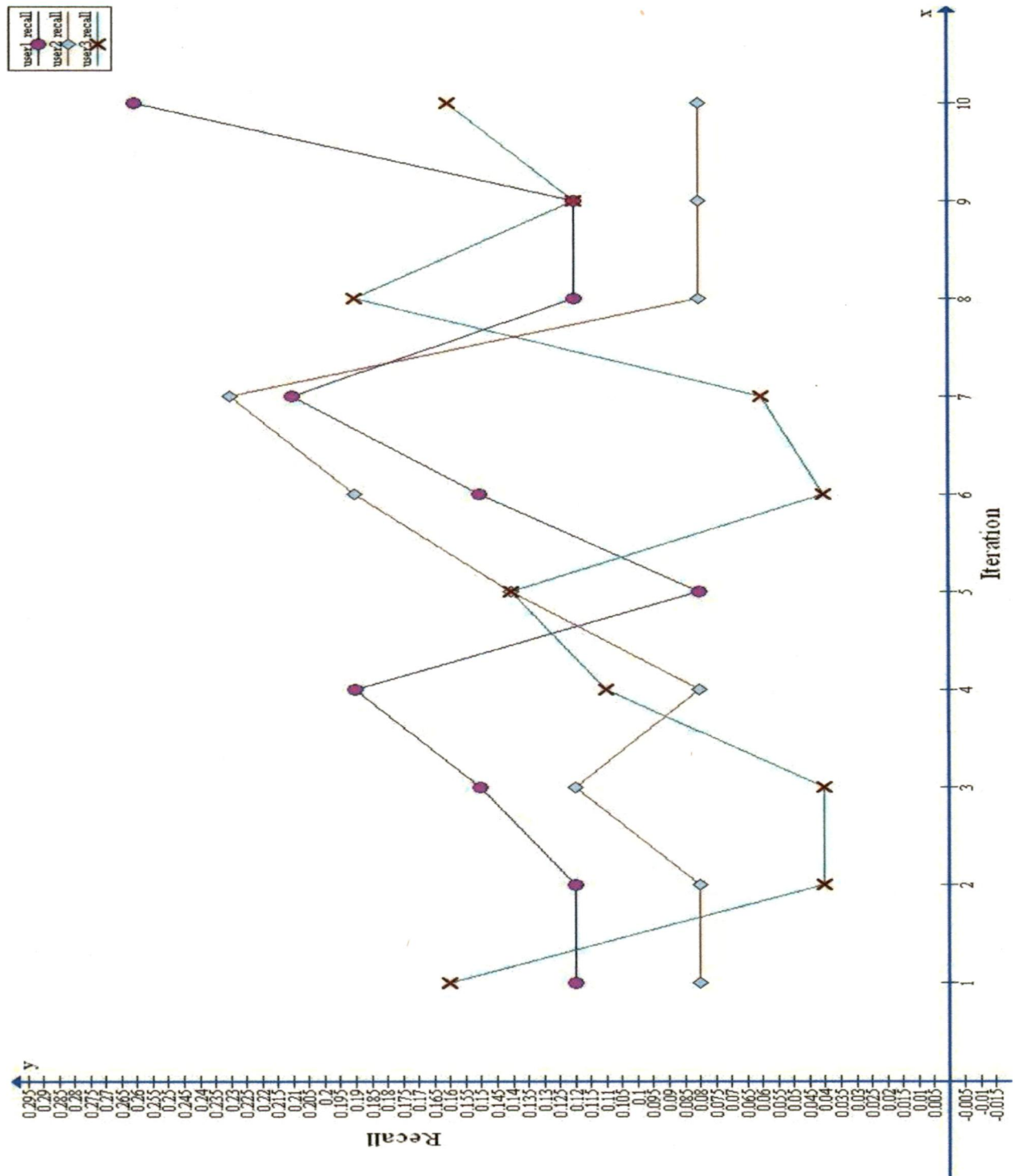
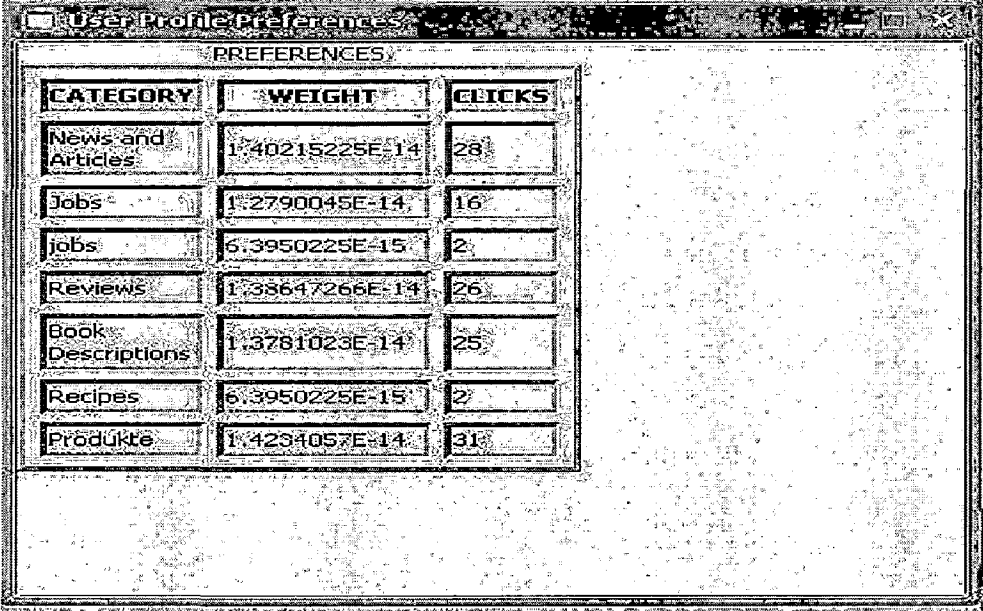


Figure 6.3 Graph for Recall

After drawing these graphs we have added a new user to the community with the following interests and found the following results.



The screenshot shows a window titled 'User Profile Preference' with a sub-header 'PREFERENCES'. It contains a table with three columns: 'CATEGORY', 'WEIGHT', and 'CLICKS'. The data is as follows:

CATEGORY	WEIGHT	CLICKS
News and Articles	1.40215225E-14	28
Jobs	1.2790045E-14	16
jobs	6.3950225E-15	2
Reviews	1.38647266E-14	26
Book Descriptions	1.3781023E-14	25
Recipes	6.3950225E-15	2
Produkte	1.4234057E-14	31

Figure 6.4 Snapshot of New user profile preferences

From the below graphs we can clearly observe that after adding a new user to the community and the user is gaining enough experience the results will affect to the other users. This result greatly affects the user with similar users (the above discussion about the similarity comes here). Non similar user's results partially affect. From the above graphs it is clear that all the affects for similar users are very promising. We can observe the graphs for user1 and user2 above those are all positive after adding the new user to the community. User3 results are partially affected.

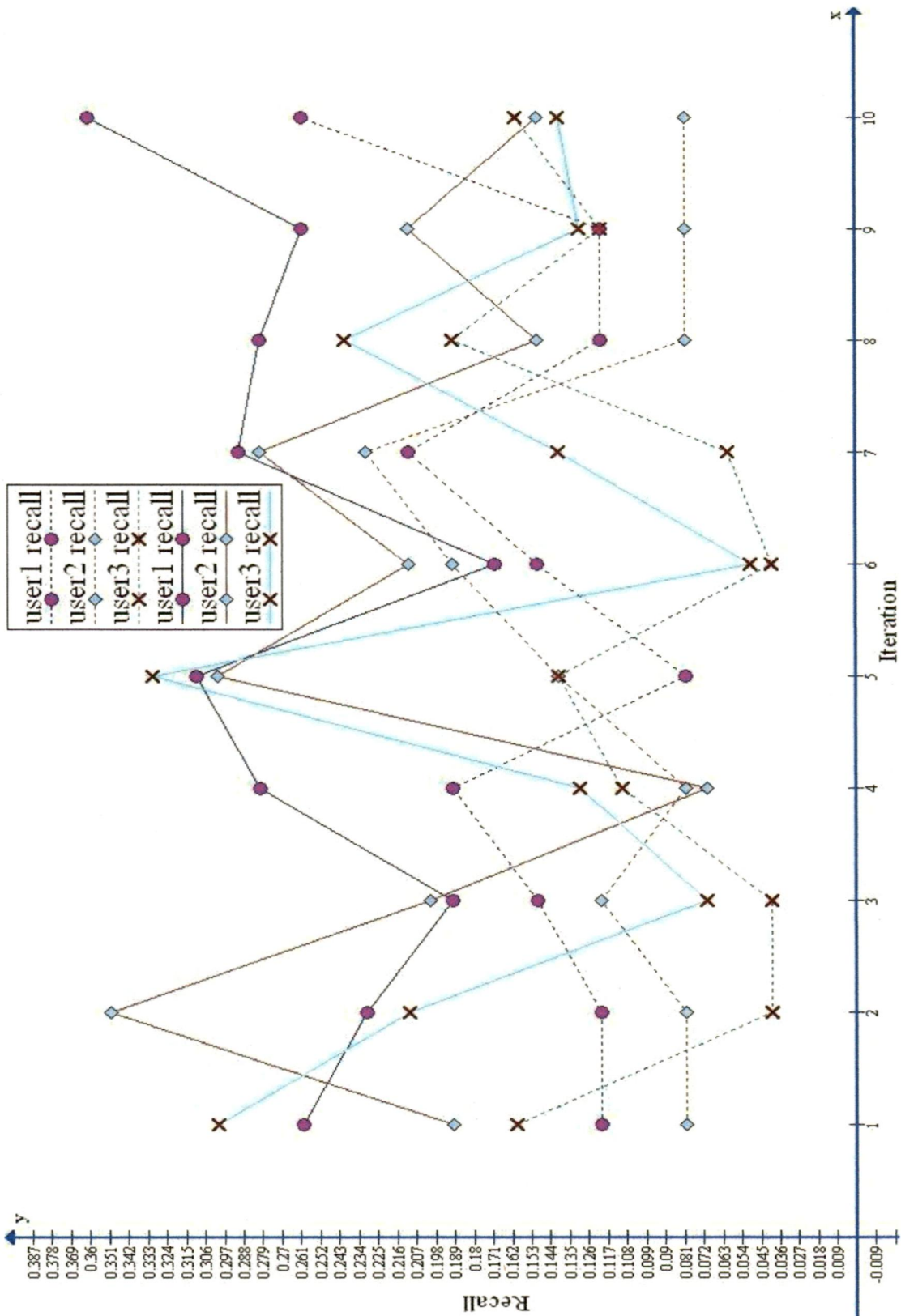


Figure 6.5 Recall Graph after introducing new user

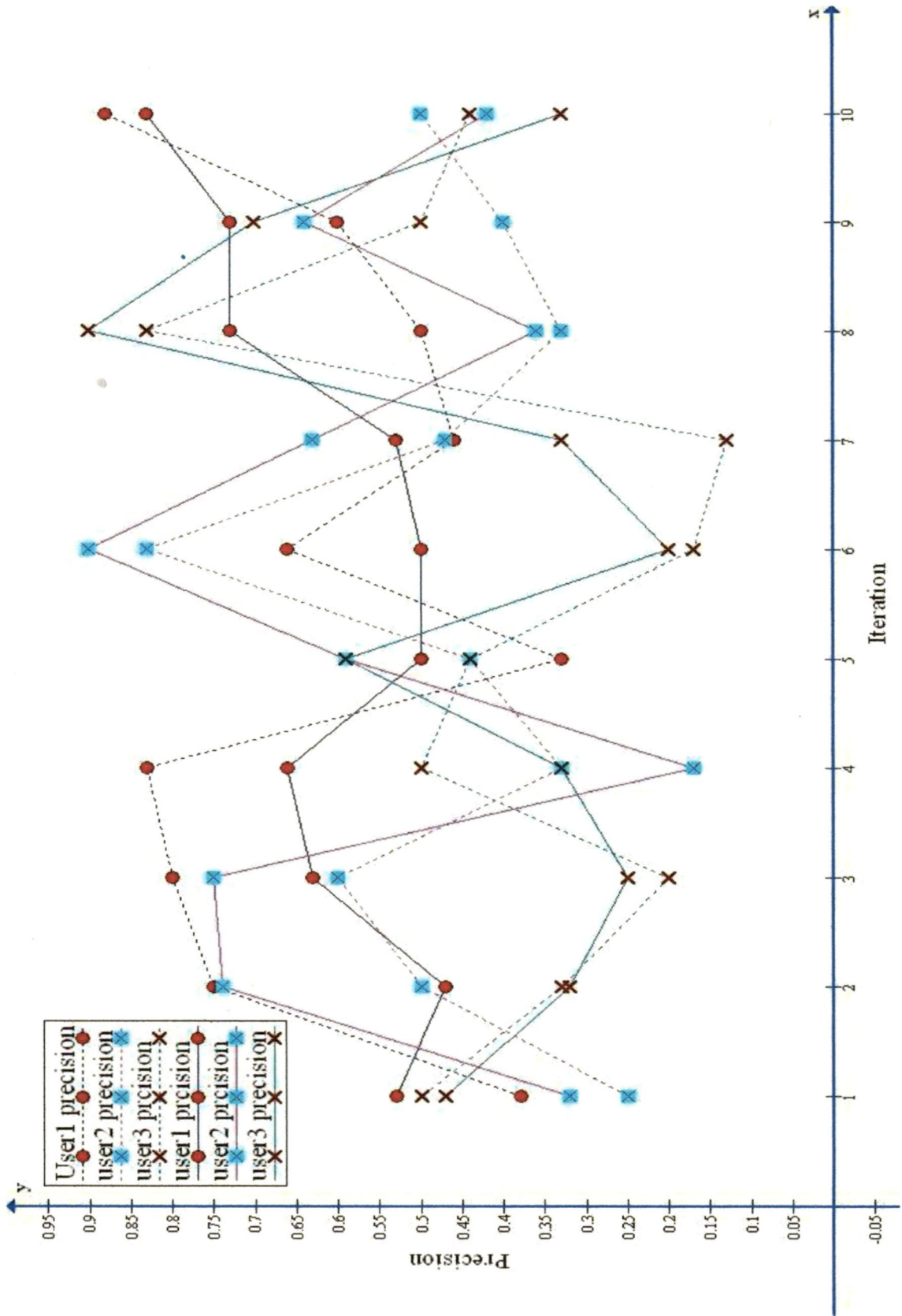


Figure: 6.6 Precision Graph after introducing new user

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

7.1 Conclusion

Two approaches for the personalized search have been studied. Merits and demerits are identified in both the approaches. An IR system model using agents and general English ontology is proposed to try to maximize the quality of search results. A sample implementation was done with some limitations and the results are analyzed in different cases. It is seen that the results are promising in comparison with the existing models.

However there are some limitations in the implementation. First one is we are separated index server and community profile it will decrease No. of results returned by the Local server. It may be possible to have some redundancy in the results because it is possible to have common document links in both Google results and community results. We are not given much importance to document weight in the sample implementation especially to re-order the results within the category. Despite of these limitations the sample implementation itself showed good improvement in the quality of the results.

7.2 Future Work

The work can be extended to reduce the storage requirement in the local server by converting the suggestions to some simplified rules. These suggestions can integrate with other document selection strategies to future improve the quality of the results. This can modified to eliminate extra software requirement at client side by making the client side implementation has a simple plug-in to the browser and deploy it from local server site.

REFERENCES

- [1] Alexander Birukov, Enrico Blanzieri, Paolo Giorgini, "Implicit, A recommender system that uses implicit knowledge to produce suggestions" Nineteenth International Joint Conference on Artificial Intelligence, IJCAI-
- [2] F.Liu, C. Yu, and W. Meng, "Personalized web search for improving retrieval effectiveness," IEEE Transactions on Knowledge and Data Engineering, vol. 16, no. 1, pp. 28-40, 2004.
- [3] Ahu Sieg, Bamshad Mobasher, Robin Burke," Learning Ontology-Based User Profiles: A Semantic Approach to Personalized Web Search ", IEEE Intelligent Informatics Bulletin, Vol .8, pp 7-18, November 2007.
- [4] James Hendler,"Agents and the Semantic Web", IEEE Intelligent Systems, pp 30-37, April 2001.
- [5] Gerald Salton, " Note About Information Science Research," Journal of the American Society for Information Science, vol. 36, no. 4, pp. 268-271,1985 .
- [6] Linda Rosen, "IT Media Lab presents the interface agents symposium: Intelligent agents in your computer?" Information Today, vol. 10, no. 3, pp. 10, 1993.
- [7] Bruce Blumberg and Galyean Tinsley, "Do the Right Things...Oh Not That!", Workshop Notes of the AAAI '95 Spring Symposium on Interactive Story Systems, Stanford University, California (March), 1995.
<http://agents.www.media.mit.edu/groups/agents/papers.html>
- [8] Pattie Maes, "Agents that reduce work and information overload", Communications of the ACM, vol. 37, no. 7, pp. 30-40,1994.

- [9] Brad Rhodes, "Pronomes in Behavior Nets. Learning and Common Sense" Section Technical Report # 95-01, MIT Media Laboratory, (November), 1995. <http://agents.www.media.mit.edu/groups/agents/papers.html>
- [10] B.Shneiderman, "Direct manipulation: A step beyond programming languages," IEEE Computer, vol. 16, no. 8, pp. 57-69.1988.
- [11] A.E. Howe and D. Dreilinger, "Savvy Search: A Meta-Search Engine that Learns which Search Engines to Query," AI Magazine, vol. 18, no. 2, pp. 19-25, 1997.
- [12] C.T. Yu, W. Meng, W. Wu, and K.-L. Liu, "Efficient and Effective Metasearch for Text Databases Incorporating Linkages among Documents," ACM SIGMOD, pp. 187-198, 2001.
- [13] M. Speretta and S. Gauch, "Personalized search based on user search histories," in Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence, WI 2005, Compigne, France, pp. 622–628, September 2005.
- [14] S. Gauch, J. Chaffee, and A. Pretschner, "Ontology-based personalized search and browsing", Web Intelligence and Agent Systems, vol. 1, no. pp 3-4, 2003.
- [15] Fabio Bellifemine, Giovanni Caire, Dominic Greenwood, "Developing multi-agent systems with JADE", Wiley publications – 2007.