COLLABORATIVE AUTONOMOUS INTERFACE AGENT FOR PERSONALIZED WEB SEARCH

BY

AHMED ALI AL NAZER

A Thesis Presented to the
DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

COMPUTER SCIENCE

JANUARY 2006

KING FAHD UNIVERSITY OF PETROLEUM AND MINERALS
DHAHRAN 31261, SAUDI ARABIA

COLLEGE OF GRADUATE STUDIES

This thesis, written by *Ahmed Ali Al Nazer* under the direction of his Thesis Advisor and approved by his Thesis Committee, has been presented to and accepted by the Dean of College of Graduate Studies, in partial fulfillment of the requirements for the degree of MASTER OF SCIENCE.

Thesis Committee

Tarek El.Busu
Dr. Tarek El-Basuny (Chairman)

Kanan Faisal
Dr. Kanan Faisal (Member)

Dr. Mamdouh Najjar (Member)

Kanan Faisal
Dr. Kanan Faisal
Department Chaiman

Dr. Mohammed A. Al-Ohali
Dean, College of Graduate Studies

١٤٢٦هـ / ١٢ / ٢٨
Date 28-1-2006

ACKNOLEDGEMENT

Acknowledgement is due to King Fahd University of Petroleum and Minerals for support of this research.

I wish to express my appreciation to *Dr. Tarek El-Basuny* who served as my major advisor. I also wish to thank the other members of my thesis committee Dr. Kanan Faisal and Dr. Mamdouh Najjar.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

THESIS ABSTRACT

FULL NAME OF STUDENT	:	AHMED ALI AL NAZER
TITLE OF STUDY	:	Collaborative Autonomous Interface Agent for Personalized Web Search
MAJOR FIELD	:	Computer Science
DATE OF DEGREE	:	January, 2006

As the Internet continues its explosive growth, the need for personalized searching tools to access the Internet is increasing since famous search engines are built to serve all users, independent of the needs of any individual user. In this thesis, we address the issue of personalization of Internet search. A system has been designed, fully implemented and named CAIA (Collaborative Autonomous Interface Agent). CAIA is developed to reside in the user’s machine not in the Internet to make all private information on his/her machine. CAIA supports communities of people in searching the Internet collaboratively by means of popular search engines. CAIA learns the user’s preferences either explicitly or implicitly from his/her browsing behavior. Those behaviors are based on a survey conducted earlier in this survey. It then stores them along with the Websites' information in a User Profile (UP). UP is containing all visited Websites' information and their relevant keywords. We mean by "relevant keywords" the keywords the user entered when s/he searches and the Websites' keywords gathered from their titles and summary. When the user enters new keywords to search for, CAIA will refine the user's query then it will start the search process. After that, it will display the search results and it will issue a process to monitor the user's behaviors on the search results. CAIA has been tested and the testing results shows a big improvement in the Internet Search using CAIA compared to famous search engines such as Google.

ملــخص الرسالــة

الاسم : أحمد بن علي بن محمد النزر
عنوان الدراسة : واجهة عميل مستقل ومتعاون للبحث الشخصي في الانترنت
التخصص : علوم حاسوب
تاريخ التخرج : ذو الحجة 1426 هـ

يتواصل الانترنت في نموه السريع وللبحث عن موقع ما يقوم مستخدموا الانترنت باستخدام آلات البحث الشهيرة مثل جوجل Google التي بنيت لتخدم جميع المستخدمين بغض النظر عن حاجة كل مستخدم، ولذا فهي لا تفي باحتياجاتهم وتعطي نتائج كثيرة جدا وبالتالي يصعب الحصول على الموقع المطلوب، وعندما يحصل المستخدم على الموقع المراد فأنه من الصعب الرجوع له إلا اذا تذكر الكلمات الرئيسية ألتى تم استخدامها من قبل في البحث، كذلك لا يمكن لأعضاء هيئة أو منظمة ما التعاون وتبادل نتائج البحث عن المواقع مع أن نسبة كبيرة من عمليات البحث متشابهة. لذا فقد زادت الحاجة إلى أدوات للبحث الشخصي التي تجلب المواقع التي يريدها المستخدم والتي تتناسب مع ميوله وخصوصياته.

في هذه الدراسة نعالج مسألة البحث الشخصي في الانترنت من خلال تصميم وتطوير نظام سميناه "واجهة عميل مستقل ومتعاون للبحث الشخصي في الانترنت" ، وهذا النظام ينصب ويعمل في جهاز المستخدم للحفاظ على خصوصية معلومات المستخدم، ويدعم هذا النظام الهيئات والمنظمات التي يرغب العاملون فيها تبادل نتائج البحث في الانترنت، ويقوم هذا النظام بتعلم خصوصيات وتفضيلات المستخدم تلقائيا من خلال سلوكه في التصفح، وهذه السلوكيات تمت دراستها من خلال استبينان قمنا به حيث استخلصنا من هذا الاستبيان سلوكيات التصفح ومن أكثرها دلالة على تفضيل المستخدم لموقع ما هي حفظ الموقع وتعدد زياراته لذلك الموقع، وقد تم اختبار هذا النظام وأثبتت التجارب التحسن الكبير الذي لمسه المستخدمون لهذا النظام للبحث عن المواقع في الانترنت مقارنة بأدوات البحث الشهيرة مثل جوجل ، فقد قلت عدد النتائج في البحث وزادت دقتها وقربها من اهتمامات المستخدم، كما أثبتت التجارب بأن النظام ذكي ويتأقلم مع سلوكيات المستخدم.

CHAPTER 1

INTRODUCTION

The Internet, the World Wide Web (WWW), is one of the largest publicly available databases of documents, and it is a good testing ground for most retrieval techniques. The Internet organizes information by employing a hypertext paradigm. Users can explore information by selecting hypertext links to other information. As the Internet continues its explosive growth, the need for searching tools to access the Internet is increasing. There are a lot of search engines that help the user in searching for documents over the Internet and play as a directory for Websites. For example, Yahoo! is a big name in Websites directories. Recently, a host of new search engine and directory sites offer a wide range of Websites-searching services [23]. Examples include Google, Alta Vista, InfoSeek, Open Text and Excite. However, these search engines are not as sophisticated as one might expect. They are built to serve all users, independent of the needs of any individual user. For example, Alta Vista presents the documents that the search engine expects one would find most relevant at the top of the list. The standard search engine ranks documents from highest to lowest based on:

- Including all of the search terms in the document.
- Including as many of the other desirable search criteria as possible in the document. Examples of desirable search criteria are: number of times the terms appear, proximity of the terms to each other, and proximity of the terms to the beginning of the document.

However, Internet users got thousands of documents when they search for certain terms and they get lost with this huge amount of documents. Moreover, they can't come back to the found document next time easily. They need to make a new search and go thorough a lot of Websites to find their desired Website.

I. PROBLEM DEFINITION

When we search in Internet for desired Websites using the famous search engines, we get thousands of documents which get us lost with this huge amount of documents. Only a limited number of those documents are relevant to us and to our interests. Moreover, we can't come back to what we found previously by certain keywords since we can't remember the keywords we entered. So, we need to make a new search and go thorough a lot of Websites to the desired Website. In addition, members of a community could not help each other in searching documents although big percentages of the search processes they perform are similar. So, the problem has turned from one of having information available to one of rapidly getting to the information that one needs.

To solve this problem, we need to customize the Internet search engines to our need and interests. Personalization becomes a popular remedy to customize the Internet environment for users. Personalization of the Internet search process is to carry out retrieval for each user incorporating his/her interests and retrieve more relevant information consistent with the user's intention. Autonomous, intelligent agents may prove to be the needed item in transforming passive search and retrieval engines into active, and personal assistants. Moreover, if those agents are collaborative, they will help the community's members to share what they found in previous searches. So, personalized collaborative autonomous intelligent information agents can help to overcome some of the limitations of communal Internet information sources such as search engines.

II. GOALS

In this work, we aim to develop a Collaborative Autonomous Interface Agent (CAIA) for personalizing the search process based on the user's interest and supporting communities of people in searching the Internet by means of popular search engines. The CAIA uses data mining and machine learning techniques in order to learn and discover user’s preferences. The CAIA resides in a user's machine for his/her trust of privacy. It is usually a running process that operates in parallel with the user, monitor user’s explicit or implicit actions from his/her browsing behavior, communicates with the many Web portals search engines (the current tool will cooperate with Google only for time being) to retrieve relevant information consistent with the user's need. The CAIA can also interact and collaborate to share knowledge about the users of the same interest and at the same time.

In general, the goals of this research are:

- To improve the information retrieval performance of Internet search engines based on specified, measurable attributes.
- To develop an autonomous, intelligent agent that will depend on Meta search engines. The agent will monitor the user's actions to prioritize the

search results. The agent will learn based on the user's preferences and information content of the search results.

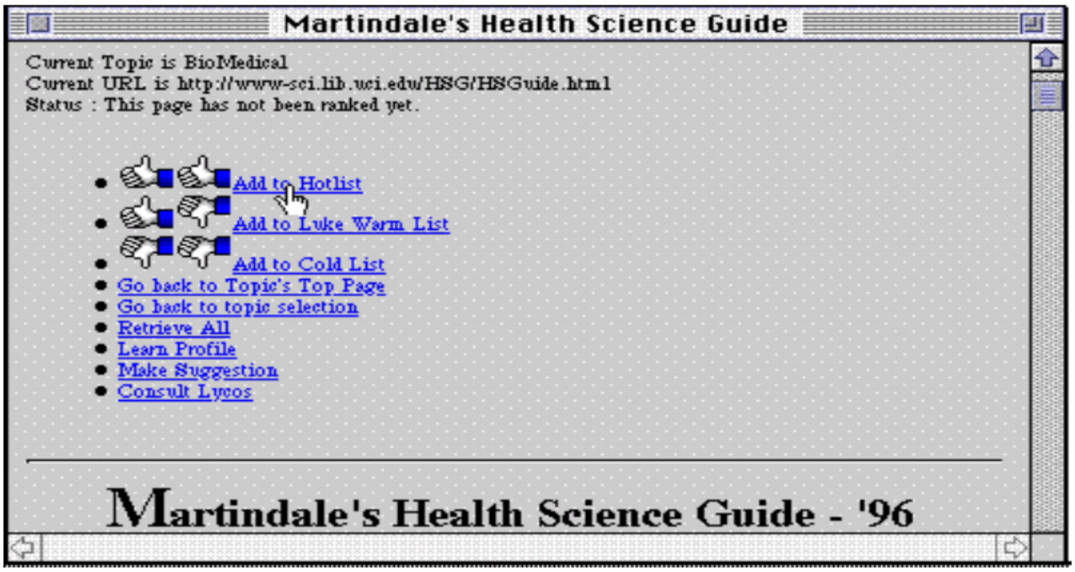
- To implement a method for the agent to learn the user's preferences during his/her searching and browsing the search results.
- To build a user's profile based on his/her preferences. This user's profile contains all visited Websites with their keywords and any entered keywords by the user in the search of those Websites.
- To filter and refine the query entered by the user.
- To filter the retrieved information based on the current user's preferences.
- To collaborate with other CAIA agents for exchanging the search results.

III. RELATED WORK

By searching the Internet and reading some published papers we found some examples of User Interface Agent for Personalization of Internet Search. The following are three examples followed by their shortcomings:

Syskill & Webert:

A software agent that learns to rate pages on the Web, deciding which pages might interest a user. It learns a user's profile based on explicit feedback of pages explored while browsing the Internet so that it can give recommendations for links and rate Lycos search results. The limitations of this agent are mainly the usage of only explicit feedback of the user and secondly the usage of only one search engine (Lycos search engine) and finally the limitation of using Netscape only as browser of Websites. More information is found in [16].


Martindale's Health Science Guide
Add to Hotlist
Add to Luke Warm List
Add to Cold List
Go back to Topic's Top Page
Go back to topic selection
Retrieve All
Learn Profile
Make Suggestion
Consult Lycos
Martindale's Health Science Guide - '96

Current Topic is BioMedical
Current URL is http://www-sci.lib.uci.edu/HSG/HSGuide.html
Status : This page has not been ranked yet.

Figure 1.1 Syskill & Webert snapshot

Letizia:

An autonomous interface agent that utilizes observed user's browsing patterns to recommend pages to view. It analyses pages in the neighborhood of the page the user is currently browsing, using limited breadth-first search. This analysis is based on page keywords and what links a user has previously followed. The agent then recommends related pages. Letizia doesn't require the user to provide an explicit feedback. It uses only implicit feedback and some examples are:

- Saving a page as a bookmark is taking as strong positive evidence
- Links skipped are taken as negative support
- Selected links can indicate positive or negative evidence, depending on how much time the user spends on the page

The main shortcomings in Letizia are firstly the depending on only implicit feedback. Secondly, it maintains persistent and slowly-changing user models. Finally, Letizia does not take care of the fact that different browsing sessions by the same user or even a single session may involve different user's interests. [20] More information about Letizia is found in [21].

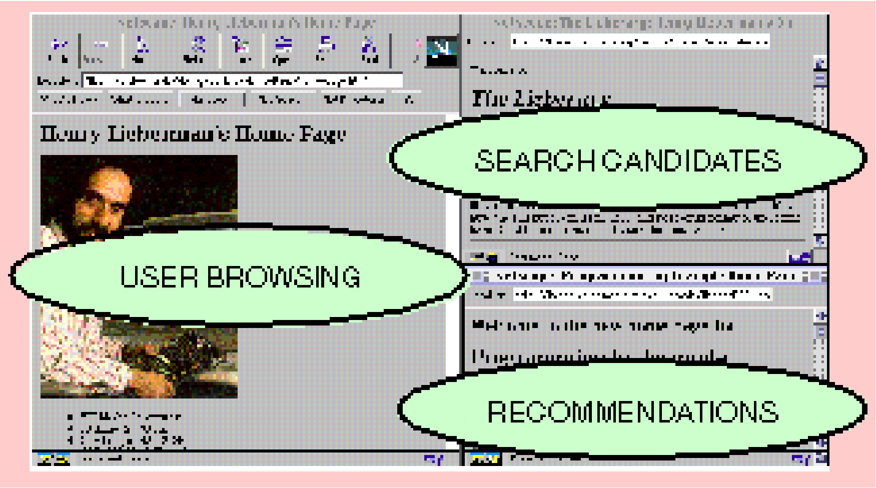
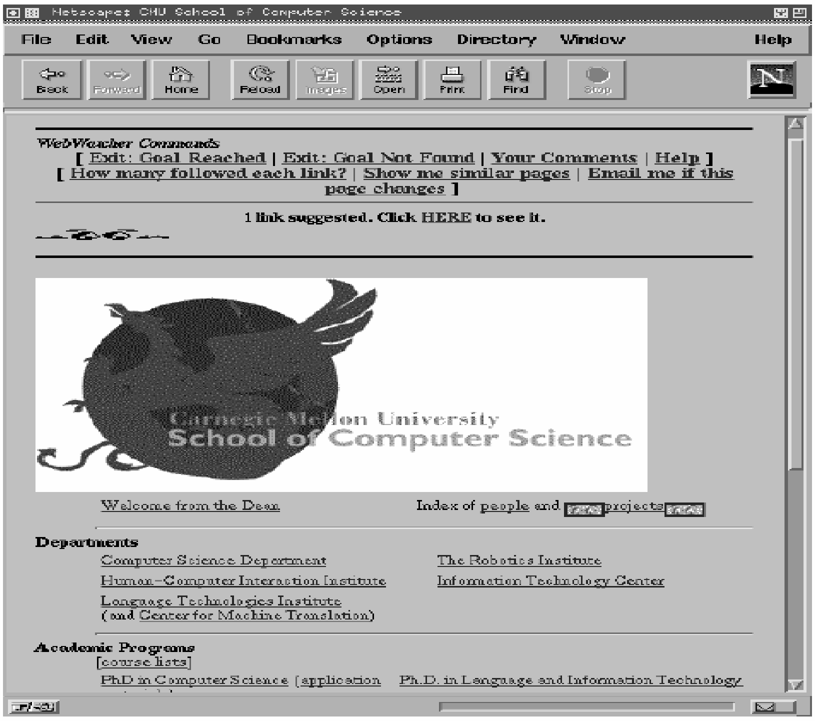


Figure 1.2 Letizia Agent

<u>WebWatcher:</u>

A goal-oriented browsing assistant that makes link suggestions by highlighting links that may lead to the pre-defined goal. WebWatcher requests an initial explicit goal from the user, and the e-mail address to keep track of the user's interests. The agent tracks user's behaviour and constructs new training examples from the encountered hyperlinks. Each hyperlink is evaluated using a utility function based on the Page, the Goal, the User and the Link.

WebWatcher has the same disadvantages of Letizia except the first one where WebWatcher is using only explicit feedback [31]. More information about Letizia is found in [11].



WebWatcher Commands

[Exit: Goal Reached | Exit: Goal Not Found | Your Comments | Help]

[How many followed each link? | Show me similar pages | Email me if this page changes]

1 link suggested. Click HERE to see it.

Welcome from the Dean

Index of people and projects

Departments

Computer Science Department

The Robotics Institute

Human-Computer Interaction Institute

Information Technology Center

Language Technologies Institute
(and Center for Machine Translation)

Academic Programs

[course lists]

PhD in Computer Science [application

Ph.D. in Language and Information Technology

Figure 1.3 WebWatcher Agent

IV. THESIS STRUCTURE

In chapter-2, we cover a literature review for the efforts in the field of personalizing the searching process of the Internet. We start with the definitions of Intelligent Agents and then we give the classification of these agents. After that, we give the definition of Interface Agent followed by classification of Interface Agents. Chapter-3 shows the survey we conducted to determine the user's behaviors priorities when s/he browses the Internet. Chapter-4 gives an overview of the design of CAIA and describes the system components. Chapter-6 gives some details of the implementation of CAIA. Chapter-6 illustrates the testing and evaluation of CAIA. Finally, Chapter 7 concludes the thesis with the benefits of CAIA and suggests some recommended future work.

CHAPTER 2

LITERATURE REVIEW

I. DEFINITION OF INTELLIGENT AGENTS

In the literature, several definitions are found for agents. James Janson in [28] gave the major ones where some of them are definitions and some are descriptions (description in terms of their tasks, autonomy, and communication capabilities):

"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, 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 computational systems that inhabit some complex, dynamic environment. They sense and act autonomously in this environment. By doing so, they realize a set of goals or tasks."

"Agents are software robots. They can think and will act on behalf of a user to carry out tasks. Agents will help meet the growing need for more functional, flexible, and personal computing and telecommunications systems. Uses for intelligent agents

include self-contained tasks, operating semi-autonomously, and communication between the user and systems resources."

Then he chooses one simple definition of agent which is "*Agents are software programs that implement user delegation*".

II. CLASSIFICATION OF INTELLIGENT AGENTS

Patty Maes [27] proposed on distinguishes between three types of agents according to the very nature of their intelligence:

- *User programmed agents*. Here, the user has to provide the "rules" to the agents so the agents follow the actions. These agents are considered to be "not very smart ."
- *Artificial Intelligence engineered agents*: These agents are considered to be smarter than the previous. They are based on traditional Artificial Intelligence Techniques.
- *Learning agents*: These agents are distinguished that they "program themselves". They learn from their users, detect their user's actions and the behavior among users. Beside that, they can learn from other agents. We mean this category by the term “intelligent agents” .

Also, agents can be divided according to the type of tasks they perform for their users. The following table demonstrates the classification of agents based on their task which is proposed in *www.BotSpot.com*:

#	Category	Sub-Categories
1	Search Bots	Image Bots, Newsgroup Bots, Meta-search Bots, On-line investigation, Music Bots
2	Shopping Bots	Auction Bots, Shopping Organizers, Shop Bots, Stock Bots
3	Tracking Bots	E-mail Notification Bots, News Bots, Spam Filtering Bots, Spy Bots, Weather Bots, Web Monitoring Bots
4	Artificial Life Bots	Adult Characters, Characters, Chatter bots, Personal Assistant Bots
5	Download Bots	Download Managers, File-sharing Bots, Off-line Browsing Bots
6	Web Development bots	Indexing Bots, Referencing Bots, Site Management Bots
7	Surf Bots	Accelerator Bots, Child Protection Bots, Form Filling Bots, Pop-up Killer Bots, Privacy Protection Bots, Spy ware Detection Bots
8	Games Bots	Build a Bot, Logic Bots, Pure Fun, Simulation Games

Table 2.1 Classification of agents based on their functions

It is mentioned in the same Website that the term *bot* has become interchangeable with *agent*, to indicate that the software can be sent out on a mission, usually to find information and report back.

III. DEFINITION OF INTERFACE AGENTS

In [26], Maes gave description of interface agents as follows: *“Instead of user-initiated interaction via commands and/or direct manipulation, the user is engaged in a co-operative process in which human and computer agents both initiate communication, monitor events and perform tasks. The metaphor used is that of a personal assistant who is collaborating with the user in the same work environment.*” The author inferred from the description that the concept behind it is to let the user to delegate some tasks to an agent as assistance. The goal is to reduce the workloads of users by making personalized agents which handle some personal work delegated by the users.

IV. CLASSIFICATION OF INTERFACE AGENTS

In [26], the author classified user interface agents based on the techniques they use such as machine learning techniques or user modeling types. He gave 10 application domain based categories and then he compared them (agent by agent) to some specific taxonomy of interface agents as follows:

1. *Character-based agents:* deal with advanced “character” based interfaces, which represent real world characters.

2. *Social agents:* talk to other agents to share information.

    - *Recommender systems*: type of social agent. They are called "collaborative filters", they find relevant items based on others' recommendations.

3. *Agents that learn about the user*

    - Monitor user behavior
    - Receive user feedback
        - Explicit feedback
        - Initial training set

4. *Programmed by user*

5. *Agents with user models*

- Behavioral model: the result of monitoring the user behavior while s/he is doing an activity.
- Knowledge-based model: the result of questionnaires and studies of users, arranged into a set of heuristics.
- Stereotypes: applied to both cases (Behavioral and Knowledge-based models), to classify the users into stereotypes (groups), in order to apply generalizations to people in those groups.

The ten applications oriented categories followed by two examples are:

1. Auction/market Domain

Kasbah is a market system in which each user has an agent. The user programs the agent with a buying behaviour profile, and the agent negotiates to buy and sell items for the user. *Sardine* is an auction agent that tries to purchase an airline ticket for the user, based on some specified preferences. The user's agent negotiates with travel agents to secure the best deal.

2. Believable/entertainment Domain

ACT is an addition to the ALIVE system. It is a creature within the ALIVE world, observing the user and learning chains of actions. It tries to help the user by completing new action chains in the pattern of previous ones. *ALIVE* is a "magic mirror" system to a 3D world. Interactive agents (such as a dog) exist for users to play with. Gesture recognition, and competing goal architecture is employed.

3. Email Filtering Domain

MailCat filters email by providing a choice of folders to the user. TF-IDF vectors are created for existing emails, and cosine similarity used to match new emails. The user has the final say, choosing one of the suggested folders or moving messages manually. *MAGI* filters emails, monitoring user behaviour and receiving relevance feedback. CN2 and IBPL are used to classify emails.

4. Expert Assistance Domain

Coach is a LISP help system that monitors user mistakes and offers unsolicited advice. A knowledge-based user model is supported, with the concept of user experience stereotypically represented. Heuristics adjust the model based on user mistakes. *Eager* automates observed repetitive HyperCard actions. It monitors the user looking for behaviour patterns, and creates helpful macros from them.

5. Matchmaking Domain

ExpertFinder monitors users' Java code and finds people who use the same classes. TF-IDF vectors represent code files, and cosine similarity is used to find similar people. *ReferralWeb* builds a social network from publicly available web pages. People's names are extracted from pages, and co-occurrence of names within pages implies a social connection. Queries such as "list docs close to Mitchell" can thus be issued.

6. Meeting Schedulers

CAP is a calendar manager, monitoring email and scheduling software to detect meeting patterns. Decision trees (ID3), using information gain to select features, are converted to production rules. *Meeting scheduling agent* schedules meetings by learning repetitive actions the user performs. Memory-based reasoning and reinforcements learning are used. Users can give explicit feedback.

7. News Filtering Domain

ANATAGONOMY is based on the Krakatoa Chronicle, providing a personalized newspaper. Implicit feedback from user activity has been added. *Butterfly* finds interesting conversations within Usenet newsgroups. The user initially provides keywords, and term frequency similarity between newsgroups and the user's profile is computed.

8. Recommender Systems

GroupLens recommends newsgroup articles based on other's ratings. Users are asked to rate every article they read, and Pearson correlation coefficient-based prediction is used to find similar users. *PHOAKS* monitors newsgroup articles, and extracts web link recommendations by a set of heuristics. A set of collective recommendations for topics is thus compiled.

9. Web Domain

CiteSeer helps users perform keyword searches for publications by using citations in documents. Heuristics extract citations, titles and abstracts, then algorithms (TF-IDF, LikeIt) classify publications based on stemmed words. Citation links lead to new search keywords. *ARACHNID* is a spider that crawls a digital library or the web, starting from the users' bookmarks, searching for user provided keywords. The user provides feedback on the pages found. A genetic algorithm is used in addition to reinforcement learning.

10. Other domains

CILA is an agent tested in an artificial, abstract domain. It tests constructive induction-based learning against AQ15c and selective induction. User monitoring, relevance feedback, initial training sets and social collaboration with other agents are supported (in its abstract world). *CIMA* is a text prediction agent, which suggests completions of sentences in a text editor. Heuristics learn from observed examples, hints and partial specifications.

CHAPTER 3

SURVEY DETERMINES FACTORS OF USER'S INTEREST IN A WEBSITE

I. SURVEY GOAL

The main goal of CAIA (collaborative autonomous interface agent) is to personalize the Web Search for the user. In order to personalize, we need to learn the user behaviours. Leaning those behaviors is not an easy task. Therefore, we thought to conduct a survey to help in learning those behaviors and also in sorting those behaviors in a way so that each behavior has some weight. I.e. Assume a user is visiting a Website, there are many possible behaviors can be done. For example, s/he can print, save, or/and add the Website to the favorite list etc. So, we aim to have default values of the weights for each factor, although CAIA gives the ability to the user to change them.

The survey was conducted by following some steps. First, a form contains the survey questions was developed and published on the Internet. Then, the form’s link was distributed among certain professional people. During two months, we have collected the responses from most of them. Finally, the results of the survey were analyzed in order to get a conclusion and come up with weights assigned to different factors that reflect the user's interest in a Website.

We used Internet to post the survey form and to collect the survey responses because it is available all the time for the respondents of the survey. Furthermore, the respondents are familiar with Internet forms so it is very easy to them to access the survey's form and do some clicks then submit the form.

II. SURVEY ENVIRONMENT

The survey was distributed among 78 persons who are high-education professionals. They all belong to Middle East culture and specifically they live in Saudi Arabia. Their ages were in the range between 25 and 45 years. They are characterized as frequently Internet users and more specifically Google search engine users. They use Google to search for any term in daily basis. The native language of them is not English but they speak English as second language. Their responses were collected within two months and they answered all questions in the survey. Furthermore, they highlighted additional factors to be considered that reflect their interest.

III. SURVEY QUESTIONS

The survey contains two questions: the first one gets an evaluation of the different factors that reflect the user's interest in a Website and the second one gets further factors if any. Respondents were asked to answer the following questions:

Question 1: "Suppose you are searching Google and you got some results. Your behavior with the results will differ and we are interested in that. Please rank the following factors which determine more your interest in a Website."

The Respondents were asked to refer to the following factors and evaluate each of them:

Print the Website

Add the Website to favorite (bookmark)

Save the Website

Time of Visit (the duration of visiting the Website)

Multiple Visits to the same Website

Explicit feedback (if it the Website asks for your feedback e.g. interested ...)

Multiple backs to a Website (going to internal links and then coming back)

Following is the scale used by subjects to compile the survey for each individual factor listed within our Web form.

Very important: means that this factor is very important and significant in determining the user's interest when s/he visits a web-page.

Important: means that this factor is just important in determining the user's interest when s/he visits a web-page.

Normal: means that this factor is neither important nor neglected so it is typical factor that does not help much in determining the user's interest when s/he visits a web-page.

Not important: means that this factor is not important in determining the user's interest when s/he visits a web-page.

Not applicable: means that this factor is neglected and it does not mean anything in determining the user's interest when s/he visits a web-page.

Question 2: "Please determine any other factors that reflect your interest in a Website."

This question permitted respondents to write-in any additional factor that helps in determining their preferences in visiting a web-page.

IV. SURVEY RESULTS

Next table reports the number of collected responses, and their percentage, in ranking the factor "Print the Website" as a factor that reflects the user's interest in a Website. Furthermore, Figure 3.1 shows a bar chart analysis of the results of that factor.

Scale	Number of responses	Percentage
Very Important	12	15.4
Important	18	23.1
Normal	22	28.2
Not Important	18	23.1
Not Applicable	8	10.3
Total	78	100

Table 3.1 Number of responses and their percentage for "print"

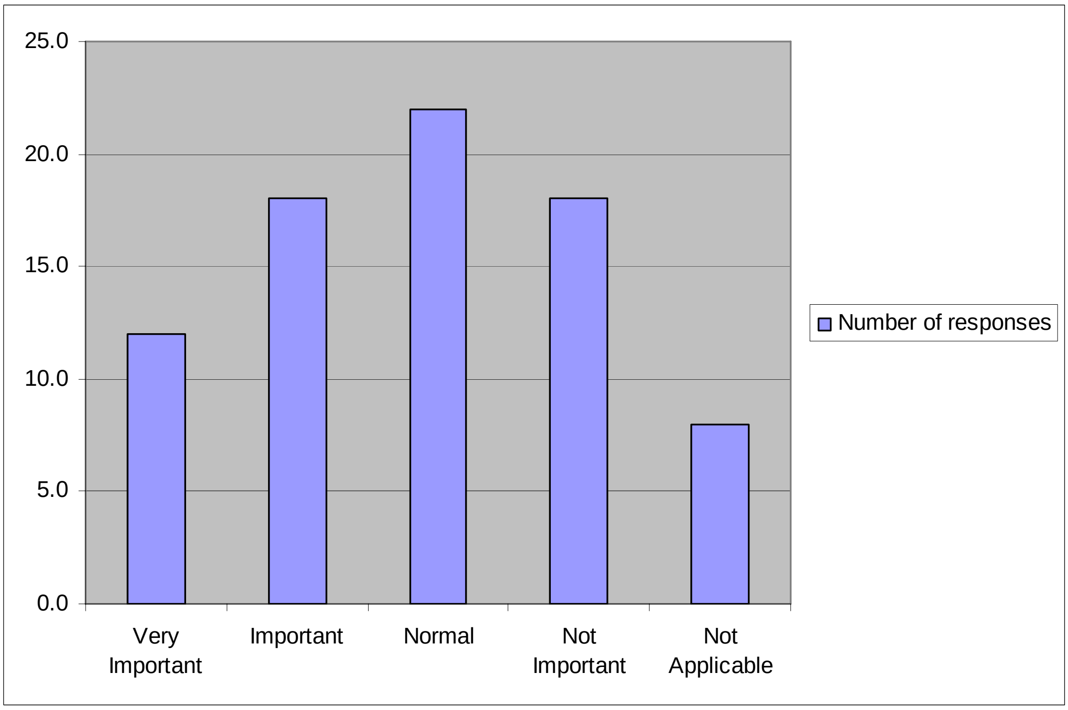


Figure 3.1 Bar chart of number of responses for “print”

Next table reports the number of collected responses, and their percentage, in ranking the factor “Add the Website to favorite (bookmark)" as a factor that reflects the user's interest in a Website. Furthermore, Figure 3.2 shows a bar chart analysis of the results of that factor.

Scale	Number of responses	Percentage
Very Important	27	34.6
Important	20	25.6
Normal	16	20.5
Not Important	13	16.7
Not Applicable	2	2.6
Total	78	100

Table 3.2 Number of responses and their percentage for “add to favorite"

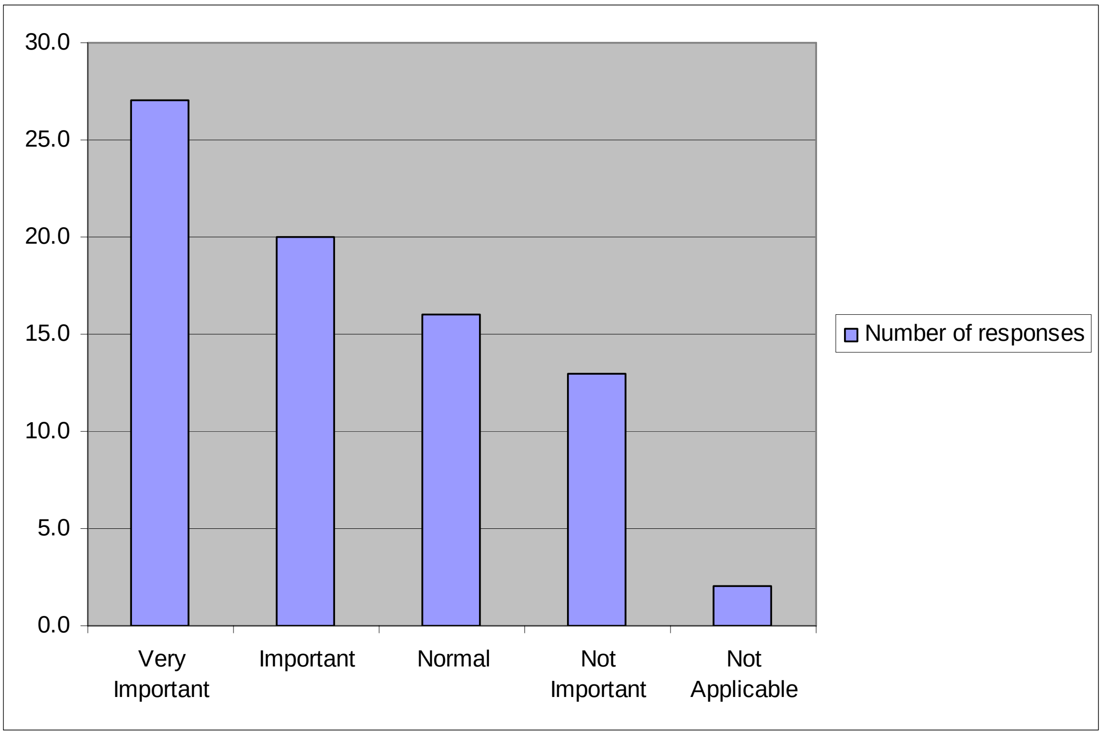


Figure 3.2 Bar chart of number of responses for “add to favorite"

Next table reports the number of collected responses, and their percentage, in ranking the factor “Save the Website" as a factor that reflects the user's interest in a Website. Furthermore, Figure 3.3 shows a bar chart analysis of the results of that factor.

Scale	Number of responses	Percentage
Very Important	22	28.2
Important	16	20.5
Normal	19	24.4
Not Important	16	20.5
Not Applicable	5	6.4
Total	78	100

Table 3.3 Number of responses and their percentage for “save"

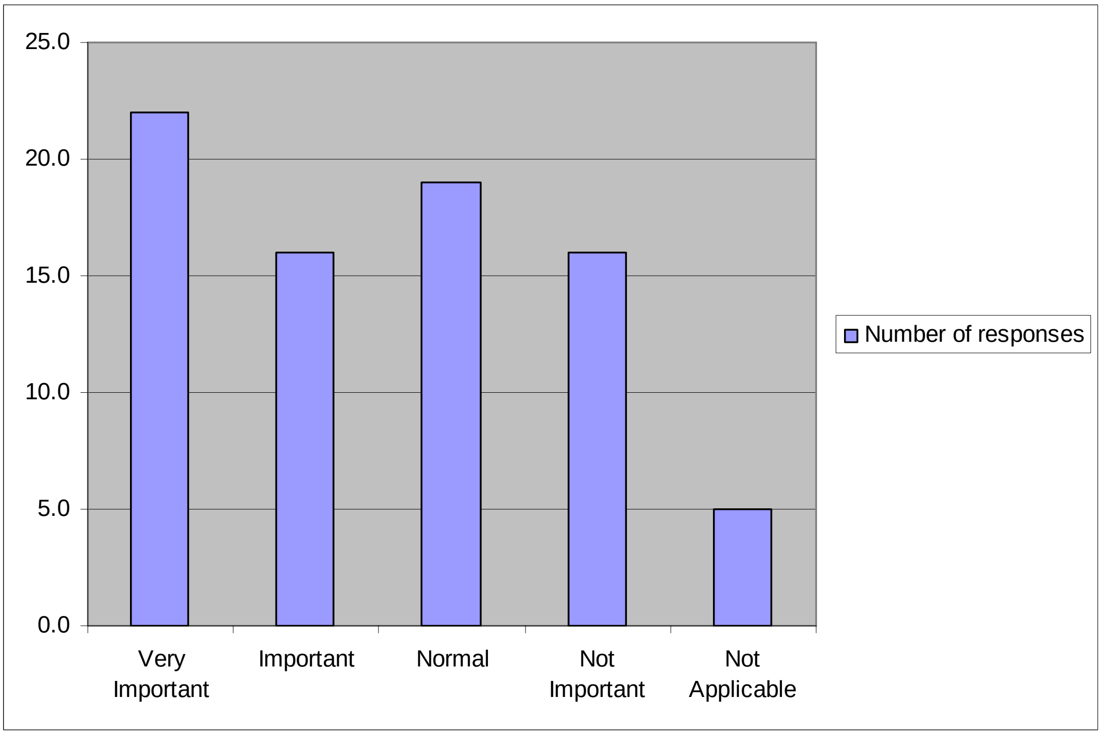


Figure 3.3 Bar chart of number of responses for “save”

Next table reports the number of collected responses, and their percentage, in ranking the factor “Time of Visit" as a factor that reflects the user's interest in a Website. Furthermore, Figure 3.4 shows a bar chart analysis of the results of that factor.

Scale	Number of responses	Percentage
Very Important	11	14.1
Important	30	38.5
Normal	22	28.2
Not Important	11	14.1
Not Applicable	4	5.1
Total	78	100

Table 3.4 Number of responses and their percentage for “time of visit"

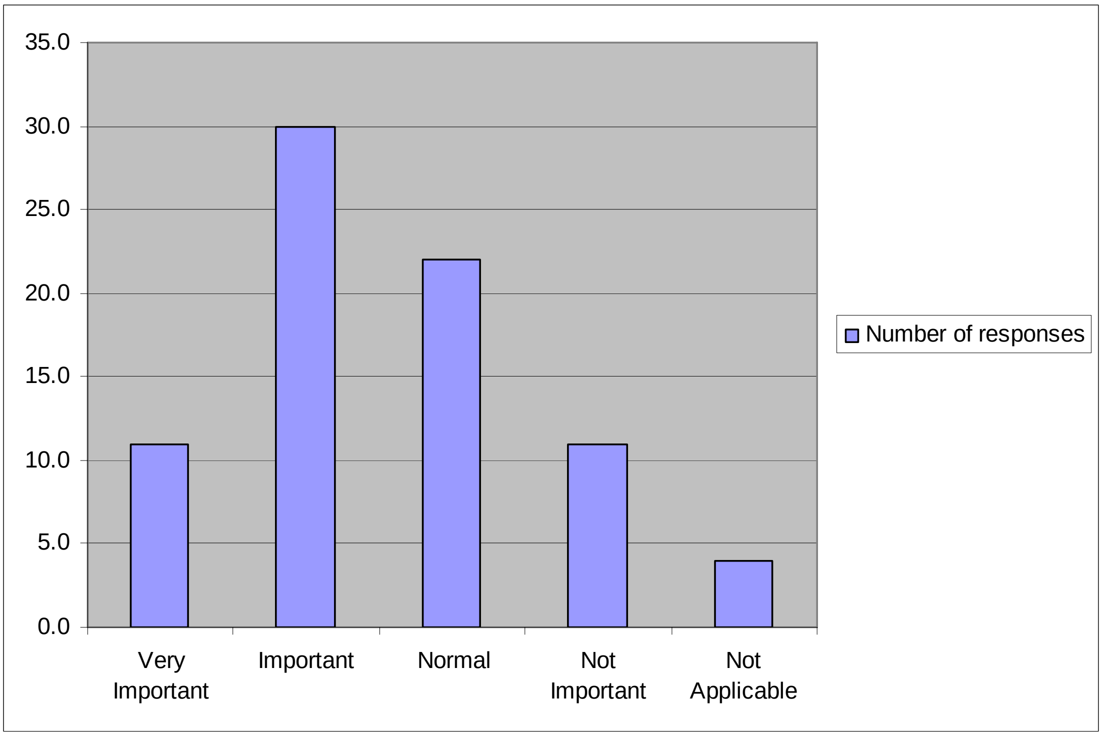


Figure 3.4 Bar chart of number of responses for “time of visit”

Next table reports the number of collected responses, and their percentage, in ranking the factor “Multiple Visits" as a factor that reflects the user's interest in a Website. Furthermore, Figure 3.5 shows a bar chart analysis of the results of that factor.

Scale	Number of responses	Percentage
Very Important	14	17.9
Important	35	44.9
Normal	18	23.1
Not Important	8	10.3
Not Applicable	3	3.8
Total	78	100

Table 3.5 Number of responses and their percentage for “multiple visits”

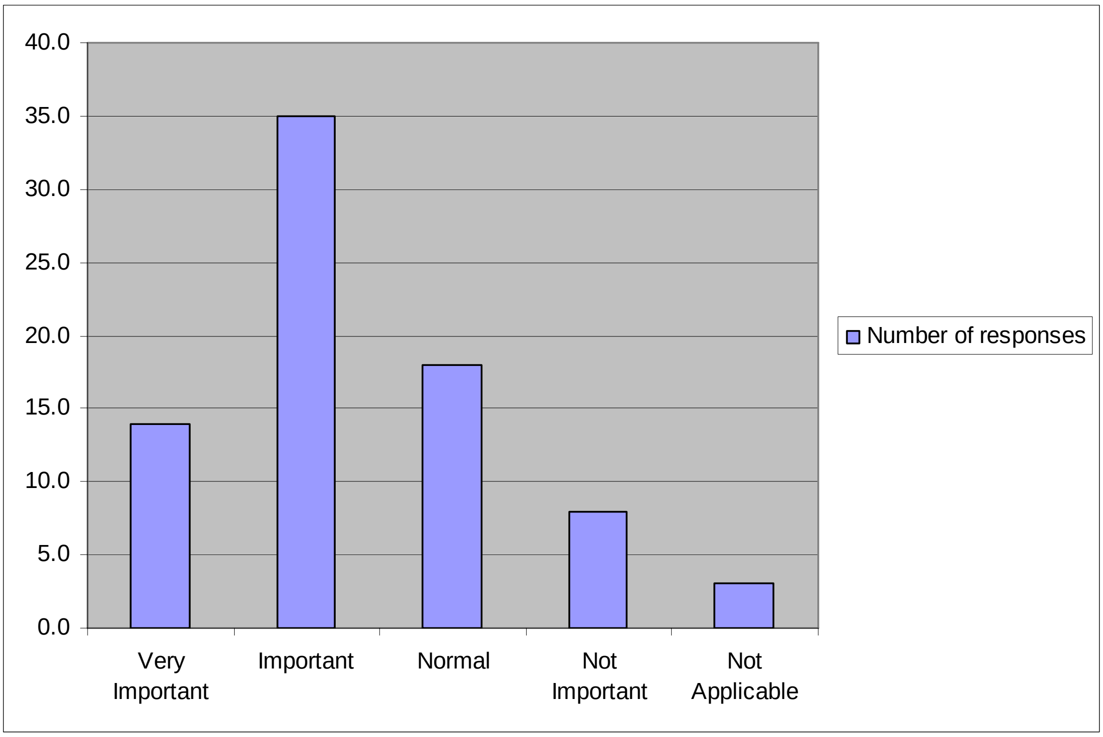


Figure 3.5 Bar chart of number of responses for “multiple visits”

Next table reports the number of collected responses, and their percentage, in ranking the factor “Explicit feedbackas a factor that reflects the user's interest in a Website. Furthermore, Figure 3.6 shows a bar chart analysis of the results of that factor.

Scale	Number of responses	Percentage
Very Important	6	7.7
Important	14	17.9
Normal	26	33.3
Not Important	27	34.6
Not Applicable	5	6.4
Total	78	100

Table 3.6 Number of responses and their percentage for “explicit feedback”

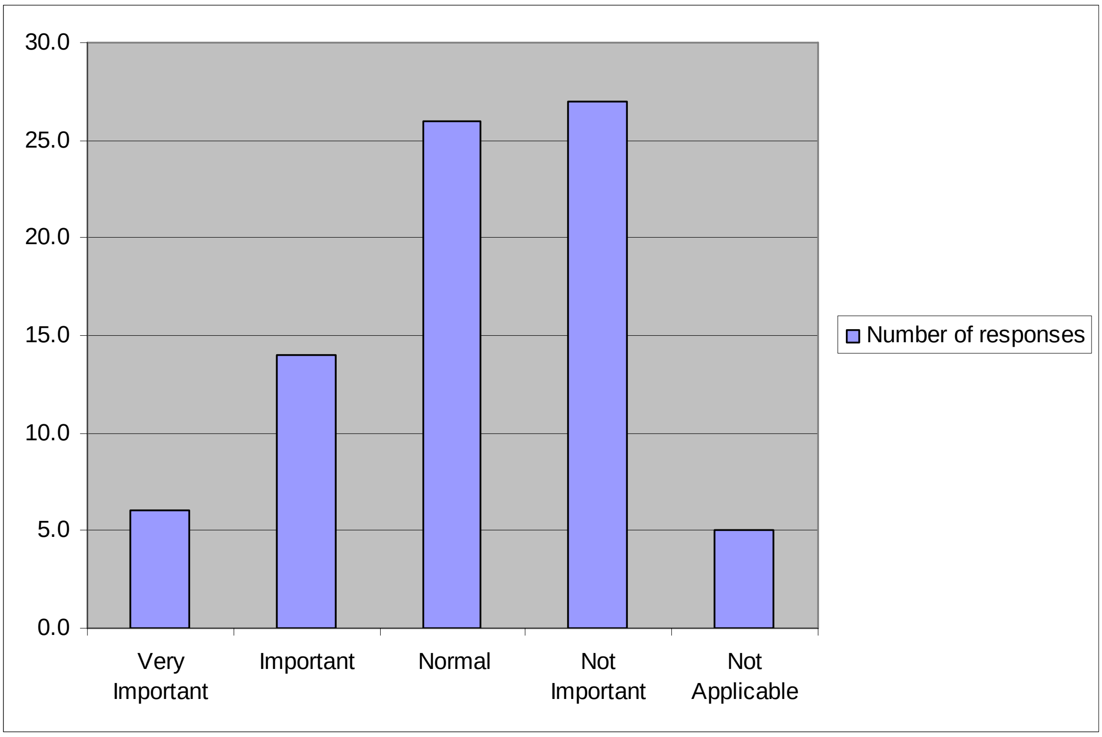


Figure 3.6 Bar chart of number of responses for “explicit feedback”

Next table reports the number of collected responses, and their percentage, in ranking the factor “Multiple backs to URL" as it reflects the user's interest in a Website. Furthermore, Figure 3.7 shows a bar chart analysis of the results of that factor.

Scale	Number of responses	Percentage
Very Important	19	24.4
Important	22	28.2
Normal	24	30.8
Not Important	8	10.3
Not Applicable	5	6.4
Total	78	100

Table 3.7 Number of responses and their percentage for “multiple backs”

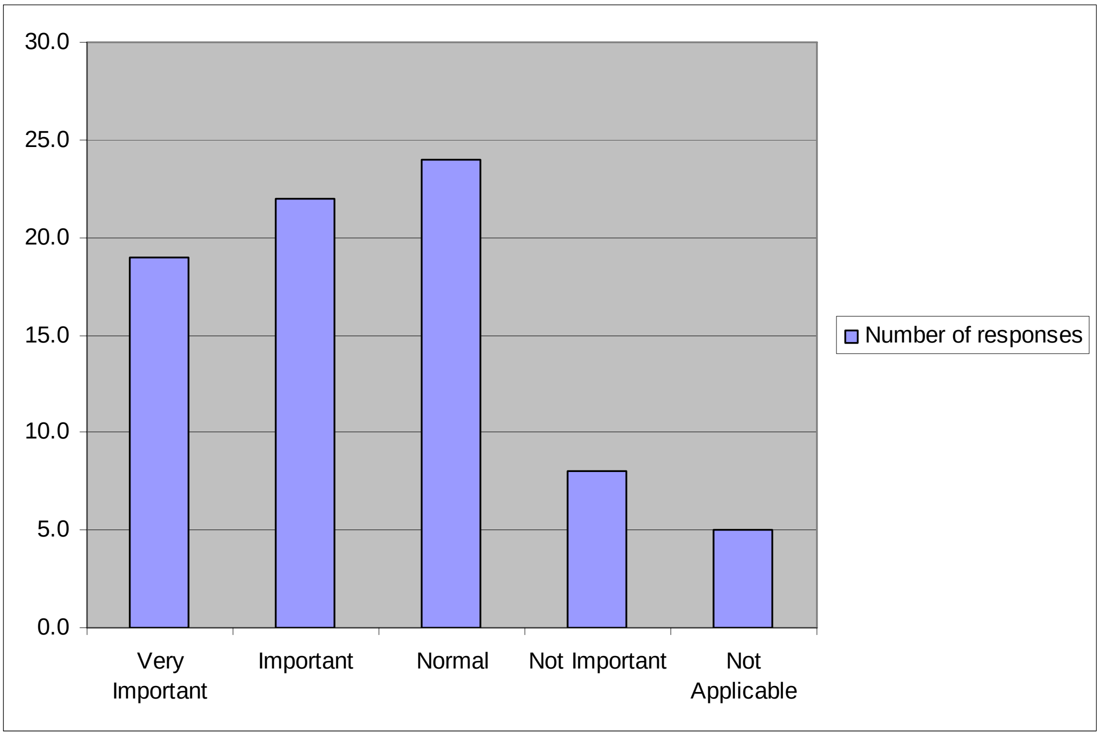


Figure 3.7 Bar chart of number of responses for “multiple backs”

Regarding the second question, we got extra suggestions that reflect the interest of a user in a Website. Thirteen extra suggestions are categorized and summarized in the following table.

Extra Factor	Number of Similar Ones
The look and colors of a Website and clear presentation of the Website.	7
The organization of the contents and the navigation of the Website.	2
The relevance to the user’s interest areas and research areas in specific.	2
Whether the Website includes a powerful search engine.	1
Loading time of the Website. The shorter the more interested.	1

Table 3.8 Extra suggested factors that reflect the user’s interest in a website

Those extra ideas should be expanded and studied further in the future work.

V. SURVEY ANALYSIS

In this section, an analysis of the survey responses is presented. The methodology of the analysis is as following. First, all the responses with their distribution are presented in one table. Then, the assumption weights of each scale in the survey are illustrated. Next, a definition of the formula that has been used to calculate the weight for each factor is given. And, the calculated weights based on that formula are presented. Finally, the result weights are discussed and analyzed.

As illustrated in previous section, total of 78 responses were collected for each factor. The following table presents the distribution of all responses for each factor.

Factor	Very Important	Important	Normal	Not Important	Not Applicable	Total
Print the Website	12	18	22	18	8	78
Add to favorite	27	20	16	13	2	78
Save the Website	22	16	19	16	5	78
Time of Visit	11	30	22	11	4	78
Multiple Visits	14	35	18	8	3	78
Explicit Feedback	6	14	26	27	5	78
Multiple Backs	19	22	24	8	5	78

Table 3.9 Results of the survey

In order to get the results of the survey, we need to calculate a weight for each factor. There were five scale choices for each factor in the survey. A weight was assigned for each scale and the analysis is based on those assumed weights. The following table shows the weight for each scale.

Scale	Weight
Very Important	1.0
Important	0.7
Normal	0.5
Not Important	-0.7
Not Applicable	-1.0

Table 3.10 Weights of each scale in the survey

Then, we calculate the average weight for each factor based on the following formula:

$$\text{Average weight} = \frac{\sum_{i=1}^{5} ai \times xi}{\sum_{i=1}^{5} xi} \qquad \text{Equation (1)}$$

Where ai is the weight for each scale i and xi is the number of responses of the scale i. So the average weight of a factor is the sum of the production of the weight of scale i by the number of responses with scale i and this sum is divided by the total number of responses.

For example, for “print” factor we observed that 12 responses “Very Important”, 18 responses “Important”, 22 responses “Normal”, 18 responses “Not Important” and 8 responses “Not Applicable”. We calculate the average weight as the following table.

Scale	I	ai	xi	ai × xi
Very Important	1	1	12	12.0
Important	2	0.7	18	12.6
Normal	3	0.5	22	1
Not Important	4	-0.7	18	-12.6
Not Applicable	5	-1	8	-8
Total			78	15

Table 3.11 Calculation of average weight for "print" factor

From the above table, we calculate the average weight for "print" factor as following:

$$\text{Average weight for “print” factor} = \frac{15}{78} = 0.1923 \approx 0.2$$

Calculation of average weight for each factor is shown in the following table.

Factor	Average Weight	Rounded Average Weight
Print the Website	0.1923	0.2
Add the Website to favorite (bookmark)	0.4859	0.5
Save the Website	0.3397	0.3
Time of Visit	0.4013	0.4
Multiple visits to the same Website	0.4987	0.5
Explicit feedback	0.0628	0.1
Multiple backs	0.4590	0.5

Table 3.12 Average weight for each factor

VI. SURVEY CONCLUSION

Based on Table 3.12, the following bar chart shows all the factors' weight.

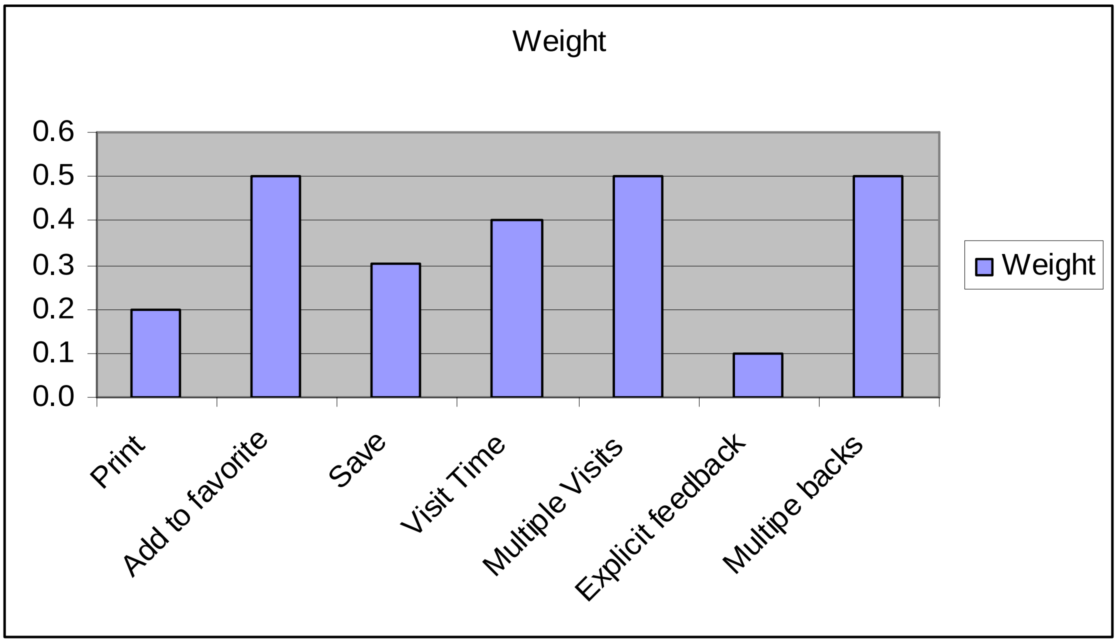


Figure 3.8 Bar chart of the average weight for each factor

Before we look to the results and the conclusions of the survey, we have to consider the survey environment mentioned in section II in this chapter.

It is observed that the "add to favorite" factor has a weight of 0.5 which is considered to be the highest weigh in the survey. That gives us a great hint on the user's interest when s/he just adds the Website to his/her favorite Websites. Adding a Website to the user's favorite list has no cost and requires a short time. Since the survey got responses from professional people who have limited time, they find this way of reflecting their interest in a Website is more effective than other ways like printing or saving. Also, "multiple visits" and "multiple backs" factors have a weight of 0.5 which is considered to be the highest weigh in the survey. It means that the user is very interested in a Website if s/he has many visits to it. Multiple backs to the same Website, i.e. going to internal links inside the website then coming back to the same website, tells us how the user is interested in this Website.

On the other hand, “Time of visit” factor have less weight, i.e. 0.4, which means it reflects the users’ interest but with less weight. It means that the time of visit is great implicit factor that determines the user’s interest but sometimes it is misleading. For example, the user visits a Website and looks inside it and tries to figure some good information but s/he did not find enough information although s/he spent good time in the Website.

“Save” factor is 0.3 which means less also. That may be because of the need of space and time to save a Website and the people who did the survey are always busy. Moreover, “print” factor got 0.2 which means that it is not weighted much in reflecting the user’s interest. It is clear that it has the same reason which is the need of extra resources like ink and papers while there are more economical ways to mark a Website as interested for the user.

Finally, “explicit feedback” factor has the least: 0.1 which means the users are not interested in making explicit feedback in normal search. They see this factor is not important to them since it asks to do some action explicitly. Users don’t want to do extra activity to reflect their inertest. They would like to deal with implicit factors rather than explicit factors.

Although we got those results displayed in Table 3.10, we can’t generalize those results to all users. That is why we put a mechanism in the developed system that allows the user to change his/her preferences weights. It depends on his age, culture, level of education, available resources, and many other factors.

CHAPTER 4

DESIGN

I. SYSTEM OVERVIEW

CAIA (Collaborative Autonomous Interface Agent) system is intended to personalize the Internet search process in order to improve it. CAIA is developed to reside in the user's machine not in the Internet to make all private information on his/her machine. CAIA learns the user's preferences either explicitly or implicitly from his browsing behavior. It then stores them along with the Websites' information in a User Profile (UP). UP is containing all visited Websites' information and their relevant keywords. We mean by "relevant keywords" the keywords the user entered when s/he searches and the Websites' keywords gathered from their titles and summary.

CAIA has three main processes illustrated in the following figure and explained later.

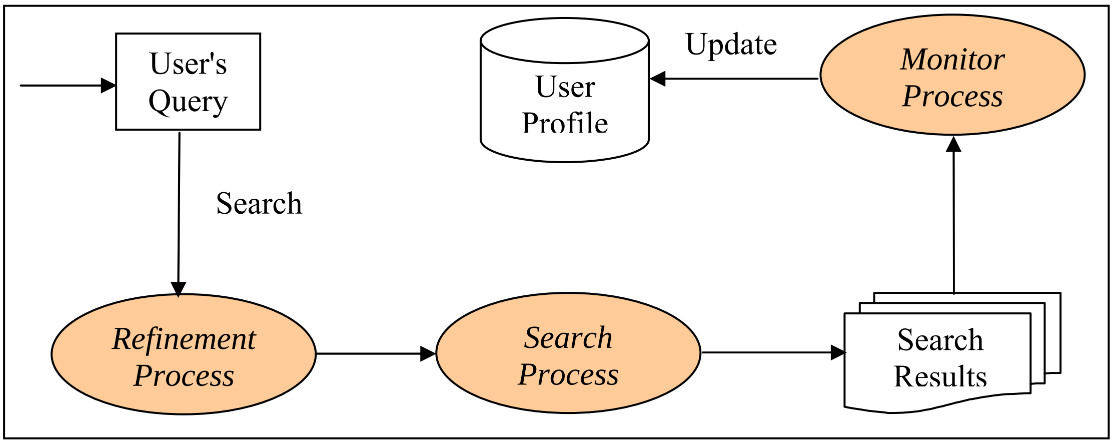


Figure 4.1 CAIA processes

When the user enters new keywords to search for, CAIA will refine the user's query then it will start the search process. After that, it will display the search results and it will issue a process to monitor the user's behaviors on the search results. So, we have three main processes done by CAIA. They are explained in the following.

Refinement Process

CAIA will start refinement process immediately after the user enters new keywords and initiate the search action. There are three main steps in the refinement process and two addition steps as following:

1. Filter the query by removing any noise words. Noise words are defined in a table in the database and can be grown based on the user’s preferences. Noise words are like suffix, prefix, (in – the, etc.).
2. Spell-check the query word by word and suggest the nearest words. We utilize Google spell checker through some APIs.
3. Refine the query by checking the UP if there are any relevant keywords to any word in the query. Then, CAIA will suggest new keywords to the user and give him/her the flexibility to either take them or ignore them.
4. Extra step done in first step in the search process when CAIA looks up in the UP. CAIA looks for thesaurus for any word in the query and then searches CAIA against those thesaurus words in addition to the original search to the user's keywords.
5. Extra step done in the second step in the search process when CAIA forward the user's query to the search engine. It looks for the most frequent word in the URLs of the relevant Websites found in first step of the search process. And

then, suggest adding it as part of the URL when forwarding the query to the search engine. For example, if the user searches for "Saudi University" and he gets http://www.kfupm.edu.sa and http://www.ccse.kfupm.edu.sa then it will suggest to him/her adding "KFUPM" as part of the URL when performing the search process using the search engine. This will improve the performance of the search process.

Search Process

After refinement process, CAIA will start the search process. There are three steps, or sources, in the search process as following:

1. CAIA will first look up in the UP, check for relevant Websites and they will be displayed if any was found.
2. CAIA will then forward the keywords to the Google search engine through a special APIs (Application Programming Interfaces). It will retrieve relevant Websites if found and display them to the user.
3. CAIA is also able to collaborate with other agents so it will forward the user's keywords to other live agents in the community. It will receive from them the relevant Websites if any and will display them to the user. Other agents will deal with any coming request with reserving the user's privacy so it will provide only the shared Websites.

Monitor User's Behaviors Process

When CAIA receives the results from different sources, the UP, the search engine and other external agents, it will re-order them based on the preferences in the UP and display them to the user. Then, it will monitor the user’s behaviors in order to determine the Websites that s/he is interested in. The user's behaviors could be noticed explicitly by allowing him to choose specific value (between 0 to 1) for the Website s/he is reading, or implicitly by monitoring the user's actions that indicate his/her interest about it; like saving the Website, bookmaking, printing, and so on. All users' behaviors will be saved in the UP to be referred to by the system every time the user searches again.

II. USE CASES

We have five main actors in the system:

1- User: the end user who will use the agent to personalize his/her search.
2- Agent: the core agent, the collaborative interface autonomous agent.
3- JADE: a middleware which is responsible for agent communication.
4- Portal: the search engine portal which has an index for a set of Websites.
5- DB: the database contains the user's profile and installs user's preferences.

We have fourteen use cases in the system distributed over two actors: User and Agent. Note that Collaboration Use Case is for both User and Agent. They are explained as follows:

Agent Use Cases

1- Search Portal: this use case is initiated by the agent to the portal which has the index of the portal pages (like a search engine). Portal will search for the keywords and will get the search results back to the agent through Process Search use case.
2- Process Search: this use case will process search results coming from the portal

through the use case Search Portal. It will reorder and filter the results according the user's profile and preferences and finally will give the search results back to the agent.

3- Process Query: this use case will process search keywords entered by the user. It will add more keywords if appropriate and suggests for better search results according the user's profile and preferences.

4- Get Data: this use case is low level function to retrieve the data from the database. The database stores all information for the user's profile and preferences.

5- Set Data: this use case is low level use case to set the data in the database. The database stores all information for the user's profile and preferences.

6- Monitor: this use case will monitor the user activities and will reflect them in the user's profile to update his/her preferences in the database. Monitor Use Case will have sensors such as print, browse, so it will reflect what it senses in the database.

7- Collaborate: this use case is initiated after the user initiated collaboration to get help from other agents. The agent will search for appropriate agents and will use JADE for communication with other agents.

User Use Cases

1- Collaborate: this use case is initiated by the user in order to get help from other agents. Then, the agent will search for appropriate agents.

2- Search: this use case is initiated by the user by entering some keywords in order to search for them in the portal. Then, the agent will take care of what the user entered, filter and search in the database and/or in the portal for best results.

3- Set Preferences: this use case is initiated by the user by setting explicitly his/her preferences to be reflected in the user's profile. This will be reflected in the database.

4- Explicit Rate: this use case is initiated by the user by explicitly rate the search results and then this will be reflected in the user's profile through the monitor use case.

5- Print: this use case is initiated by the user by printing a document (from the search results) which reflects his/her interest and this will update his/her user's profile through the monitor use case.

6- Add to Favorite: this use case is initiated by the user by adding a document (from the search results) to his/her favorite list. This reflects his/her interest and this will update his/her user's profile through the monitor use case.

7- Browse: this use case is initiated by the user by browsing a document (from the search results) to his/her favorite list. This reflects his/her interest and this will update his/her user's profile through the monitor use case.

The following figure illustrates all the use cases of CAIA.

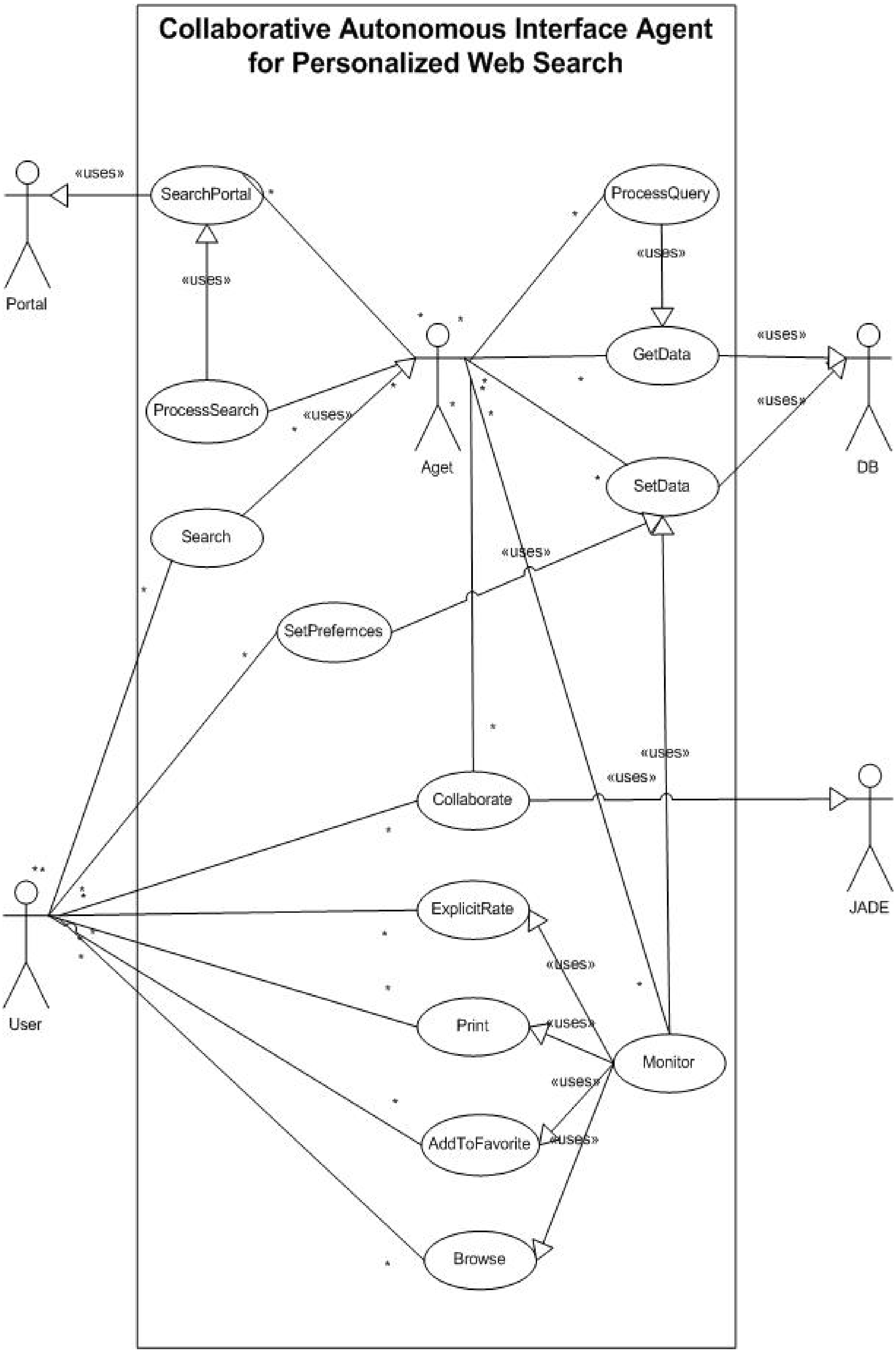


Figure 4.2 Use cases of CAIA

III. SYSTEM COMPONENTS

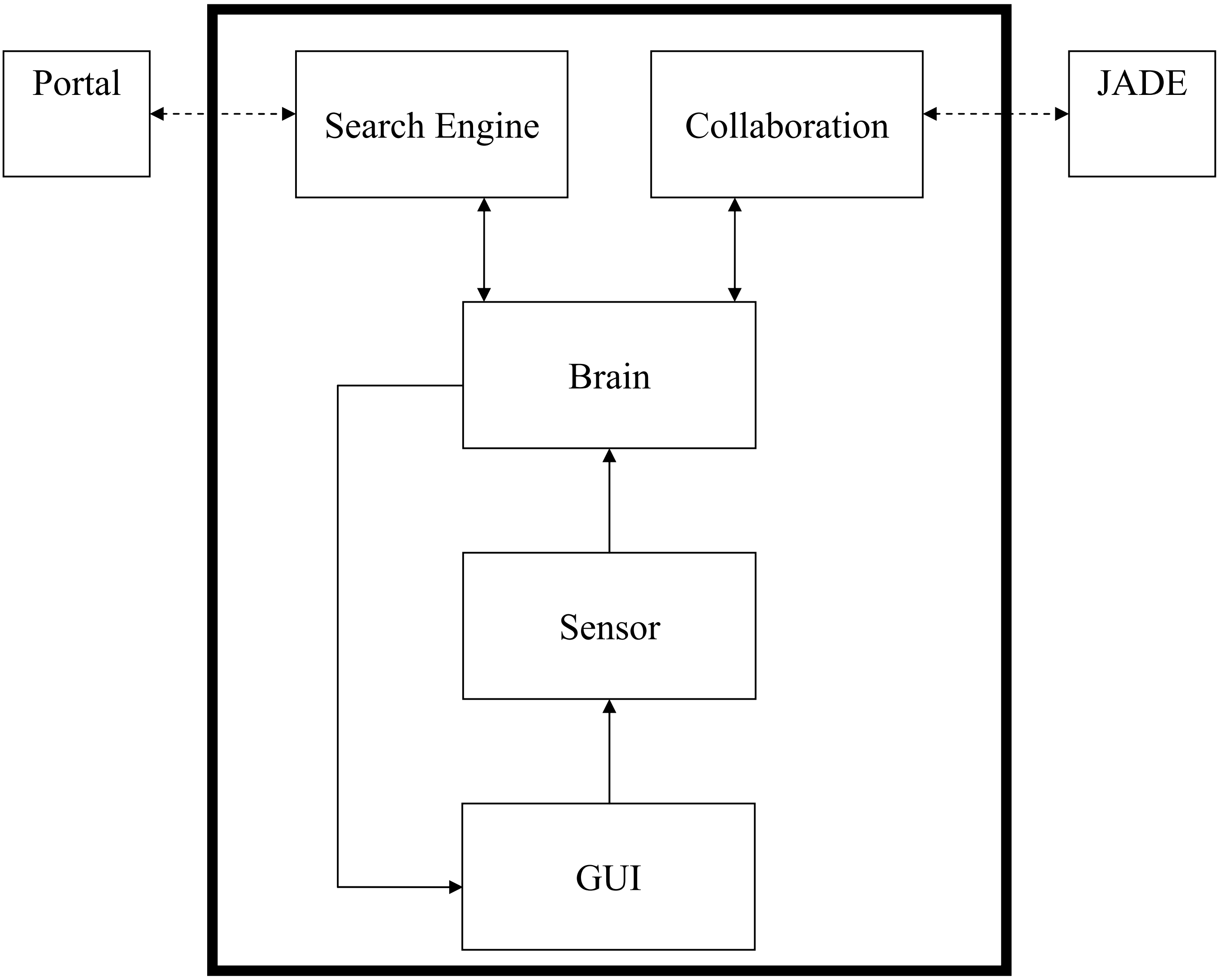


Figure 4.3 CAIA components

There are five main components in the system and two outside the system. They are explained as follows:

1. *Search Engine Component*: it is responsible for searching the search engine portal for any search request comes from the *Brain Component*. *Search Engine*

Component deals with the portal and gives the portal the filtered search keywords in order to get the search results. After getting the search results, the *Search Engine Component* forwards them back to the *Brain component* where they will be filtered and re-ordered.

2. *Brain Component*: it is the core component of the system and responsible for:
    1. Rebuild user's query (which entered in the *GUI Component*) and put extra keywords to get better search results.
    2. Filter search results according the user's profile, after getting them from the *Search Engine component*.
    3. Get the user activities through *Sensor Component* which senses all users' activities and reaction with search results.
    4. Initiate collaboration request to Collaboration Component to enhance the search results for the user's query. Also, it looks for appropriate agents to the user which can help him/her.
    5. Control the user's profile (i.e. the database) and the preferences.
3. *Collaboration Component*: it is responsible for collaboration between agents. It deals with JADE to communicate with other agents and get information to help in the search. It will be responsible for the protocol that each agent understands.
4. *Sensor Component*: it is responsible for sensing user's activities and reporting them to the *Brain Component* which processes them and reflects it in the user's profile. It deals with *GUI Component* to retrieve any action done by the user.
5. *GUI Component*: it is the end user interface. It provides the user a mechanism to search keywords and display search results then give him/her a floor to explicitly give his/her feedback. It will have the following functionalities:

i. Get user’s search keywords, and feeds it to the *Brain Component*.
ii. Display search results which are take from the *Brain Component*.
iii. Browse any Web page of the search results and forward any user’s action to the *Sensor Component*.
iv. Get user’s explicit feedback for any Web page and forward it to the *Brain Component*.
v. Give the user’s options to save, print, add to favorite and forwards any user’s action to the *Sensor Component*.
vi. Give the user’s options to search for another agent using *Collaboration Component*.
vii. Give the user’s options to set his/her preferences and forward it to the *Brain Component*.

6. *Portal Component*: it has the index of the Website. It might be Meta Search Engine or portal database. Regardless, it will take keywords and will search for Websites match the keyword and finally give back the search results. We use the word "portal" for the standard search engine regardless with what we use in the implementation.

7. *JADE Component*: it is responsible for low level communication between agents. It deals with *Collaboration Component* and looks for live agents. Then, it gives back the list of available agents. Then, if there is a request to get information from the external agent, it will get it from them. We use JADE as the middleware in the communication between agents because it is reliable and globally used in all agents.

IV. BRAIN COMPONENT IN FOCUS

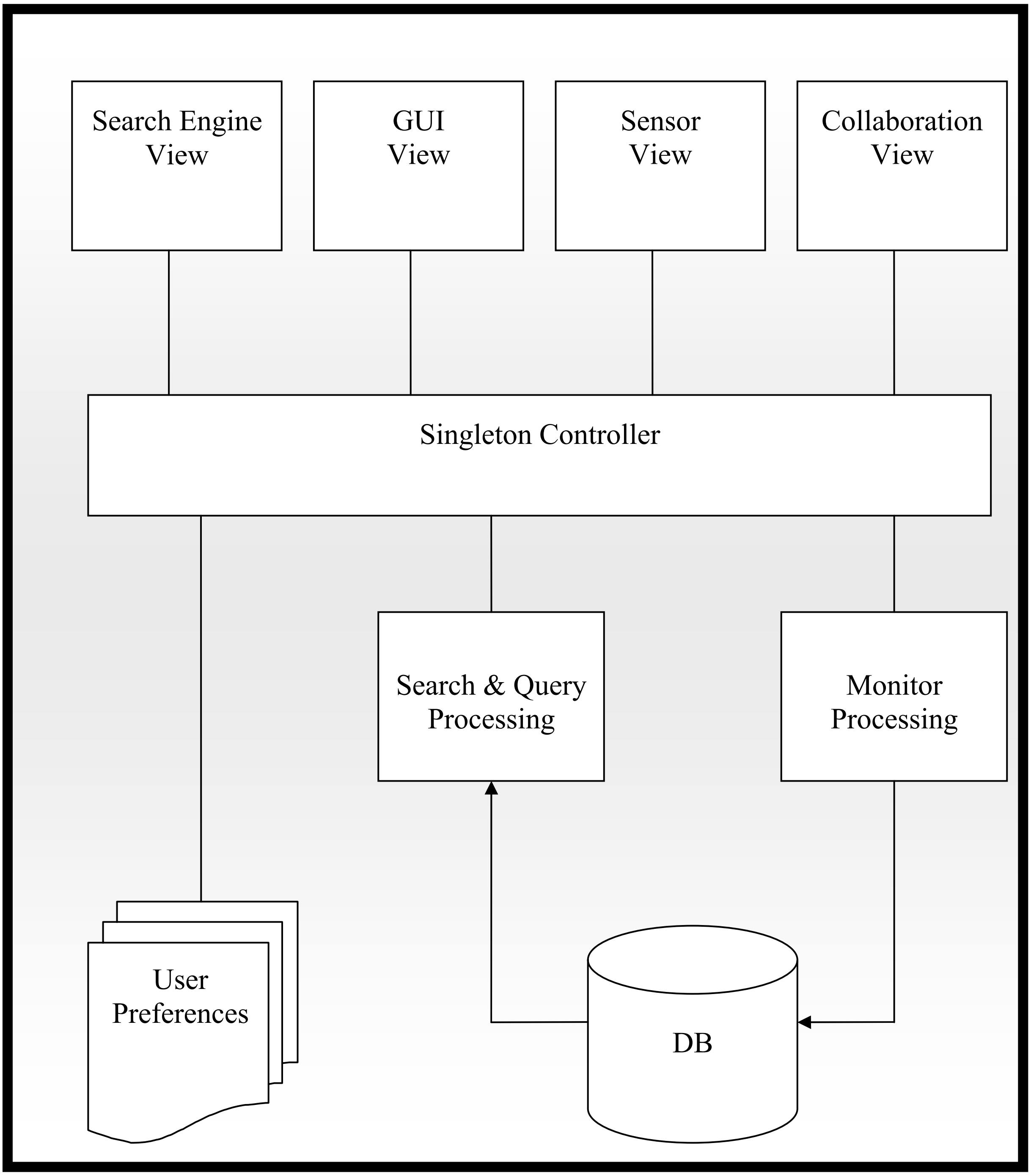


Figure 4.4 Brain component design

Since the *Brain Component* is the core component in the system, we focus on it in this section. The design of *Brain Component* is following MVC paradigm (Model, View, and Control) since it is reliable, easy to understand and it is the standard nowadays. There will be a view for each component talks with the *Brain Component*. We will have the following views:

1- *Search Engine View*: This class is interfacing with *Search Engine Component* and responsible to provide a refined query to the search engine and handle the search results got from the *Search Engine Component*.

2- GUI View: This class is interfacing with *GUI Component* and responsible for getting the search query from the user, refine it and return the search results.

3- *Sensor View*: This class is interfacing with *Sensor Component* and responsible for handling any event done by the user after displaying the search results.

4- *Collaboration View*: This class is interfacing with *Collaboration Component* and responsible for deliver search results for a certain query got from different agent. Also, it is responsible for determining the user's preferences for other agents through the *Collaboration Component*.

Also, *singleton controller* will be responsible to interface with the controllers of the views. We will have two controllers:

1- *Search & Query Processing*: This class is interfacing with *Search Engine View*, *GUI View* and *Collaboration View*. It is responsible for filtering and refining the user's query. Filtering will remove the noise words from the query while refining will add more keywords from the database. This class is also responsible for issuing the search from different sources, database, portal and external agents that have similar preferences to the user's agent. Then, it is filtering the search results based on the user's profile in the database, such as re-ordering the results based on the keywords weights.

2- *Monitor Processing*: This class is interfacing with *Sensor View*. It is responsible to handle all events done by the user after displaying the search results. It updates the database, the user's profile, to reflect his/her actions and reactions to the search results.

Finally, we have two data models:

1- *The database (User's Profile):* This data model will store the user's profile for any visited Websites and any entered keywords. It also represents the weight of different keyword whether it was entered by the user or got from the visited Websites.

2- *User's Preferences File:* This data model will store the user's preferences which represent the user's personality and interesting. For example, it saves the preferred weight for each user's behaviors. Also, based on the user's preferences, *Collaboration Component* will find relevant agents.

The next sections will explain each model in details.

V. DATABASE DESIGN

CAIA system has a relational database which represents the User’s Profile (UP). It holds all historical data for the user such as the visited Websites and the search queries. The database contains five tables named: *Keywords*, *URL*, *URLKeywords*, *Events* and *NoiseWords*. The first four tables are related with relationships and the last, *NoiseWords*, is not related to all other tables.

Keywords table holds the keywords which are the words that help us to identify Websites. *URL* table holds the Websites' URL (i.e. the Internet address such as http://www.kfupm.edu.sa which is URL for KFUPM Website). *Keywords* and *URL* tables are related with a many-to-many relationship. i.e. a URL can have more than one keyword and a keyword can have more than one URL. For example, KFUPM Website has the keywords: King, Fahd, University, Petroleum, Minerals and KFUPM while CCSE in KFUPM has keywords same as KFUPM keywords and more CCSSE, Computer, Engineering, etc. We observe that a URL has more than one keyword and a keyword has more than one URL. So, we created a table, *URLKeywords*, to hold their primary keys, *keywordID* and *URLID*, and the weight for each keyword that related to a certain URL.

Events table holds all events that were done to the URL, such as printing, saving etc. *URL* table is related to *Events* table with a one-to-many relationship as one URL obtains many events. That is why we keep the *URL* table's primary key as reference key in *Events* table.

Finally, *NoiseWords* table holds all noise words that are used in query refinement process. The following Figure gives a complete design schema of the database followed by an explanation for each table individually.

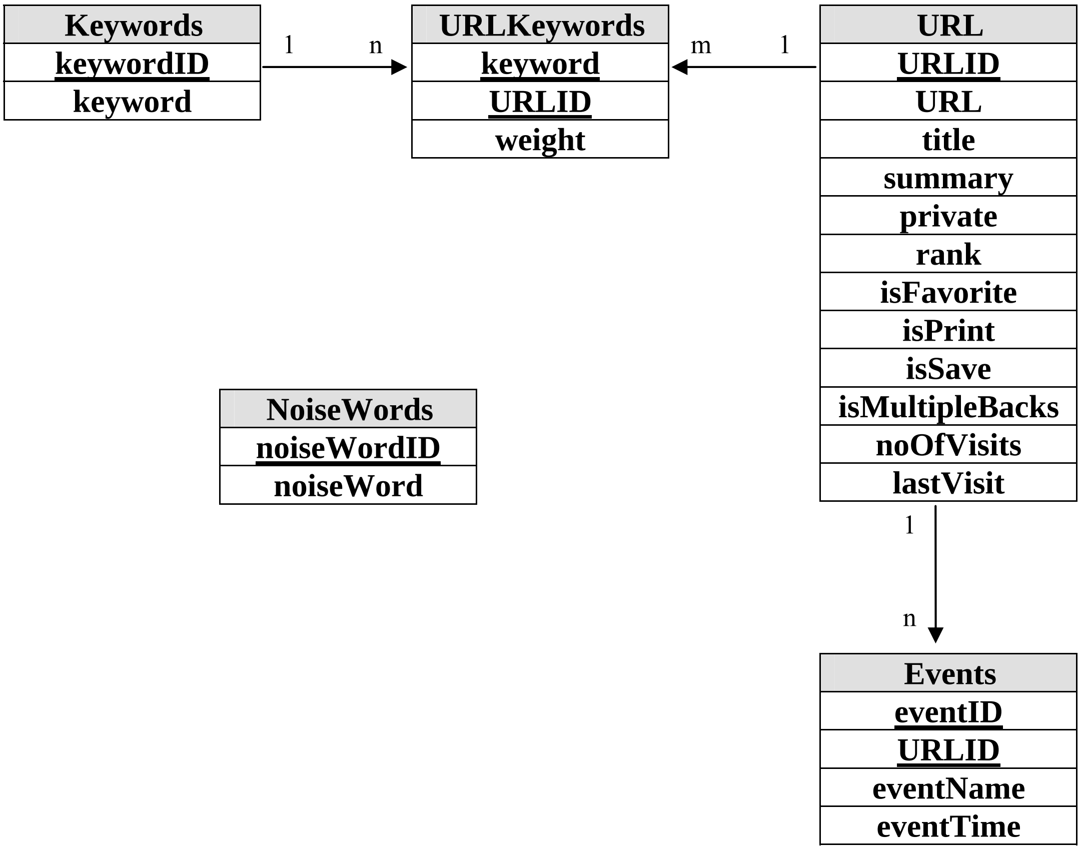


Figure 4.5 Database design schema

URL Table:

URL table holds the Websites' URL and all other attributes related to that URL. For example, it holds *title* and *summary* attributes which are related to the URL. In addition, it has some attributes related to the properties of the URL. For example, it has *private* and *rank* attributes which are attributes of the URL. Finally, *URL* table holds some attributes related to the events obtained on the URL. For example, it holds *noOfVisites* and *lastVisit* attributes which help us in getting that information quickly rather than accessing *Events* table and search for last record and count all related records.

Attribute	Data Type	Description
URLID	Number(integer)	Primary key which identify different URLs
URL	Text	The link of the web page with 255 characters in length.
title	Text	The title of the web page with 255 characters in length.
summary	Memo (Text)	The summary of the web page with up to 65,535 characters length.
private	Number (0/1)	Identifier if the URL is private so it is not shared to other agents.
rank	Number(1...5)	Rank of the URL which is set explicitly by the user.
isFavoirte	Number (0/1)	Identifier if the URL is added to the favorites list.
isPrint	Number (0/1)	Identifier if the URL is printed.
isSave	Number (0/1)	Identifier if the URL is saved.
isMultipleBacks	Number (0/1)	Identifier if the URL is visited by multiple backs with the same URL.
noOfVisits	Number (0/1)	Counter for the number of visits of the URL.
lastVisit	Text	255 Character contains the date of last visit of the URL.

Table 4.1 Fields of *URL* table

Keywords Table:

Keywords table holds the keywords which are the words that help us to identify Websites. It has an identifier which is *keywordID* and the keyword itself as a string of characters.

Attribute	Data Type	Description
keywordID	Number(integer)	Primary key which identify different keywords.
keyword	Text	A keyword entered by the user or got from a specific URL with 255 characters in length.

Table 4.2 Fields of *Keywords* table

URLKeywords Table:

URLKeywords table holds *URL* and *Keywords* tables' primary keys, *keywordID* and *URLID*. It also holds the weight for each keyword that related to a certain URL.

Attribute	Data Type	Description
keywordID	Number(integer)	Foreign key point to the keyword id.
URLID	Number(integer)	Foreign key point to the URL id.
Weight	Number (double)	The weight of the keyword for a specific URL.

Table 4.3 Fields of *URLKeywords* table

Events Table:

Events table holds all events that were done to the URL, such as printing, saving etc. It holds the *URL* table's primary key as reference key. It also holds the event's attributes such as the event's name and the event's time.

Attribute	Data Type	Description
Eventide	Number(integer)	Primary key which identify different event.
URLID	Number(integer)	Foreign key point to the URL id.
eventName	Text	The name of the event (255 characters). For example, print, start visit etc.
eventTime	Date/Time	The time in which the user do the event on the URL.

Table 4.4 Fields of *Events* table

NoiseWords Table:

NoiseWords table holds all noise words that are used in query refinement process. It has an identifier, *noiseWordID*, and the *noiseWord* filed as string of characters.

Attribute	Data Type	Description
noiseWordID	Number(integer)	Primary key which identify different noise words.
noiseWord	Text	Contains the noise word with 255 characters long.

Table 4.5 Fields of *NoiseWords* table

VI. USER’S PREFERENCES DESIGN

The user’s preferences file reflects the user’s personality and preferred choices. It is the place where the user can specify his/her favorite settings: implicit and explicit settings. An example of explicit settings is whether s/he wants CAIA to help him/her in suggesting relevant keywords or not. Another example of implicit settings is how much "printing the website" weights in calculating the whole weight of a keyword in a URL.

User’s preferences help also in searching and finding relevant agents if the user would like to collaborate with other agents in the community. It gives the user the flexibility for both sides of collaboration: for getting information from other agents and for providing information to other agents. Sharing information with other agents will guarantee the privacy of the user so it will not share any private URLs.

User’s preferences can be settled by the user in first use of the agent and it can be updated later at anytime. All settings will be reflected directly when the user’s update his/her preferences.

User profile contains two major parts:

1. Settings of the weight scale for each factor.
2. Settings of advance options.

Settings of the weight scale for each factor:

We mean by a factor, whether it is implicit or explicit factor, a set of behaviors that identify the interest of a Website, such as printing and saving the Website. Based on Chapter-3, we had a survey and we identified a set of those factors and we gave each factor a weight based on this survey. We set those weights as the default and we give the user the flexibility to change them based on his preferences. In these setting options, we give the user a similar form to the survey's form in order to set his/her preferred weight for each factor. Each factor will have a value which is reflected in the weight of the keyword. The value is ranged between 1 and 5 Value 1 is the most interest and value 5 means no interest. The design of this part is as the following table.

Attribute	Data Type	Description
Print	Number (1…5)	This defines the user preference in “Print” factor.
Favorite	Number (1…5)	This defines the user preference in “Favorite” factor.
Save	Number (1…5)	This defines the user preference in “Save” factor.
TimeOfVisit	Number (1…5)	This defines the user preference in “Time of Visit” factor.
MultipleVisits	Number (1…5)	This defines the user preference in “Multiple Visits” factor.
ExplicitFeedback	Number (1…5)	This defines the user preference in “Explicit Feedback” factor.
MultipeBacks	Number (1…5)	This defines the user preference in “Multipe Backs” factor.

Table 4.6 Design of first part of user preferences

Settings of advance options:

"Advance options" contains list of option that gives the flexibility to the user in using CAIA. It also gives him/her the options whether to collaborate with other agents or not.

All of those options are explained in the following table.

Attribute	Data Type	Description
SpellChecking	True/false	This determines whether the user wants to use spell checking.
SuggestRelatedKeywords	True/false	This determines whether the user wants to let CAIA suggests related keywords or not.
MonitorAllBrowsing	True/false	This determines whether the user wants to let CAIA monitors all browsing activities.
MonitorSeachResults	True/false	This determines whether the user wants to let CAIA monitors all activities after search.
SearchGoogle	True/false	This determines whether the user wants to let CAIA searches using CAIA+Google component.
SearchExternalAgents	True/false	This determines whether the user wants to let CAIA searches with the help of external agents.
SearchMyAgent	True/false	This determines whether the user wants to let CAIA searches with the help of CAIA agents.
Days	Number	This determines whether the user wants to let CAIA searches with the help of CAIA agents within "Days" range.

Table 4.7 Design of second part of user preferences

CHAPTER 5

IMPLEMENTATION

The system was fully implemented and named CAIA (Collaborative Autonomous Interface Agent). CAIA was implemented in the Java programming language using the Java 2 SDK. 1.4. The system can be described as follows:

1. Three tier architecture: GUI Component, Sensor Component and Brain Component. Each component is considered as a stand alone sub-system.
2. Extendibility: the system is designed very well in such away adding a new major functionality is just matter of plug-in.
3. Portability: the system is implemented using JAVA language which is platform-independent.

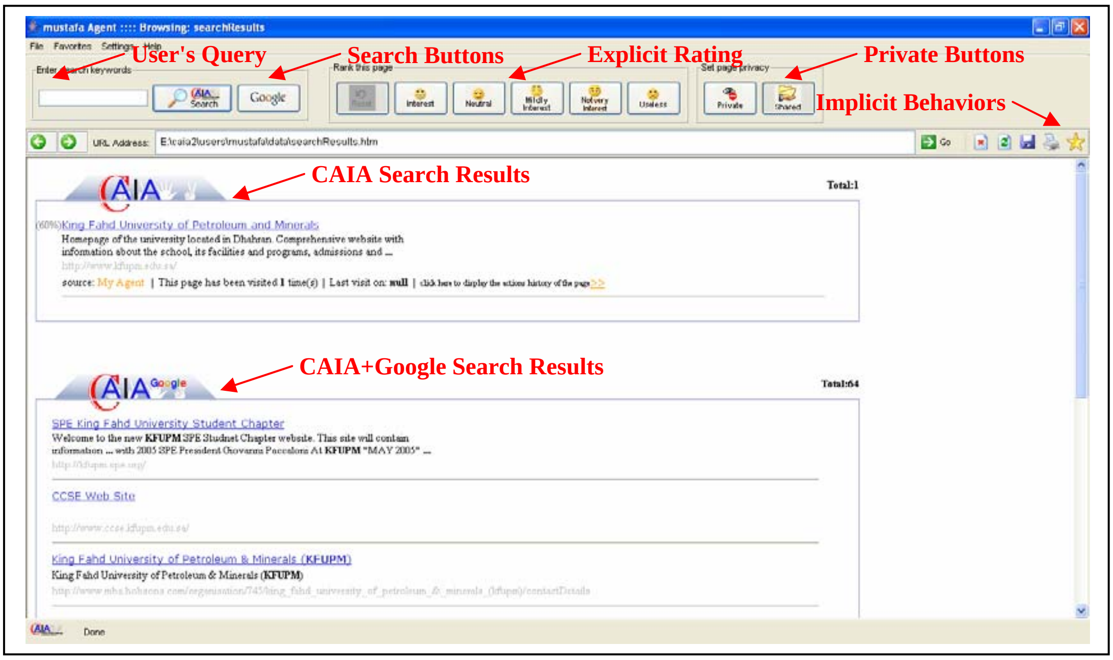
We have utilized some third party packages and enhance some of their functionality in our system. The following third party packages were used:

1. *JADE (Java Agent DEvelopment Framework)* [31]: it is a software framework fully implemented in Java language. It simplifies the implementation of multi-agent systems through a middle-ware that claims to comply with the FIPA (The Foundation for Intelligent Physical Agents) [35] specifications. We use JADE for easily communicate between agents.
2. *Google Web APIs (beta)* [32]: The Google Web APIs service gives us query access to Google's web search , enabling us to develop software that accesses billions of web documents that are constantly refreshed. We use these APIs to

get optimal search results for the user's query. Also, we use the Google's spell-checker API to spell-check the user's query.

3. *The Wintertree Thesaurus Engine class library* [33]: The Wintertree Thesaurus Engine Java SDK contains a Java class library that can be used to add a general-purpose thesaurus capability to your Java applications. We use this class library to get thesaurus for the words to make the search more practical.
4. *EZ JCom* [34]: EZ JCom provides a bridge between Java programs and COM (ActiveX) objects. Using EZ JCom, Java programs can call COM objects, can receive events from COM objects, and can even embed the COM objects inside Java user interface elements. We use EZ JCom to embed Internet Explorer in our system and to listen to the events the user do in the Explorer.

Details about the implementation of the main components of the system are described in the following sections.



(60%)King Fahd University of Petroleum and Minerals
Homepage of the university located in Dhahran. Comprehensive website with information about the school, its facilities and programs, admissions and ...
http://www.kfupm.edu.sa/
source: My Agent | This page has been visited 1 time(s) | Last visit on: null | click here to display the actions history of the page>>

SPE King Fahd University Student Chapter
Welcome to the new **KFUPM** SPE Student Chapter website. This site will contain information ... with 2005 SPE President Giovanni Paccaloni At **KFUPM** "MAY 2005" ...
http://kfupm.spe.org/

CCSE Web Site
http://www.ccse.kfupm.edu.sa/

King Fahd University of Petroleum & Minerals (KFUPM)
King Fahd University of Petroleum & Minerals (**KFUPM**)
http://www.mba.bohanna.com/organisation/745/king_fahd_university_of_petroleum_&_minerals_(kfupm)/contactDetails

Figure 5.1 Snapshot of CAIA

I. THE GUI COMPONENT

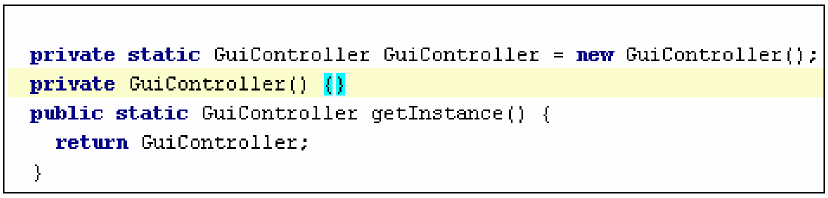
GUI Component is responsible for the direct interaction between the agent and the user. It is implemented in a separate package called *gui*. The implementation corresponds to the design described in Chapter-4. Table-5.1 summarizes Java classes of the *gui* package. The more detailed explanation of important classes follows in the next sections. Each class is saved in a file of the same name with the extension ".java".

Class name	Task of the class
BrowserPanel	Partial panel from the main frame. It is used to display the Search Results and Website's Content.
CAIAGUI	The main class in the GUI component which initializes the main frame.
Favorite	Object container of the favorite list.
FavorieLink	Object container of the favorite link.
FilesSettings	Object container of the files' settings.
FrameOrganizeFavorites	Favorites' organizer frame that displayed to the user.
FramePreferences	Preferences' settings frame that displayed to the user.
FrameProxy	Proxy's settings frame that displayed to the user.
GuiController	GUI Component's classes controller.
MainFrame	The main GUI Component frame.
PreferencesSettings	Object container of the preferences' settings.
PrintUtilities	Print's settings frame that displayed to the user.
ProxySettings	Object container of the proxy's settings.

Table 5.1 Classes of the *GUI Component*

GUI Controller

GUI Controller is a singleton class that is available all the time to control and serve other entities in the component.



```
private static GuiController GuiController = new GuiController();
private GuiController() {}
public static GuiController getInstance() {
  return GuiController;
}
```

Figure 5.2 Code shows how to make the class singleton.

GUI Controller has other functionalities described in the following.

i. It has a bean of the main frame object (*MainFrame*) with a setter and getter methods to serve other classes which need to get the main frame.

ii. It has a bean of the search result object (*SearchResult*) with a setter and getter methods to serve other classes which need to get the search results.

iii. It has a bean of the files' settings object (*FilesSettings*) with a setter and getter methods to serve other classes which need to get the files' settings. It initializes and creates an object of (*FilesSettings*) which holds the paths of all the files used by the system. It is initiated only once by the *GUI Controller*.

iv. It has a method: "*getConfirmedUserKeyword()*" which is called by the *Brain Component*. It is responsible for displaying a dialog to the user with the suggested keywords and gets the confirmed query.

v. It has a method: "*getConfirmedMostFrequentWord()*"which is called by the *Brain Component*. It is responsible for displaying a dialog to the user with the suggested most frequent word in the relevant Websites' URLs. It gets the confirmation from the user either to use it in the search process or not.

vi. It has a method: "*deserializeFavoriteObject()*" which is called by the *MainFrame* once the system starts. It reads the user's favorites from his/her *Favorite File* and then de-serializes it into *Favorite* object.

vii. It has a method: "*serializeFavoriteObject()*" which is called by the *MainFrame* once the user organizes his favorites' list, e.g. deleting some favorite Websites. It sterilizes *Favorite* object with the latest update and writes it back into the user's *Favorite File*.

Main Frame

The main frame includes and initiates all sub-frames, panels, dialogs and buttons. It also registers the required items into the Sensor-GUI view which is acted as the mediator of the GUI. Figure 5.3 shows the design of the *Main Frame*.

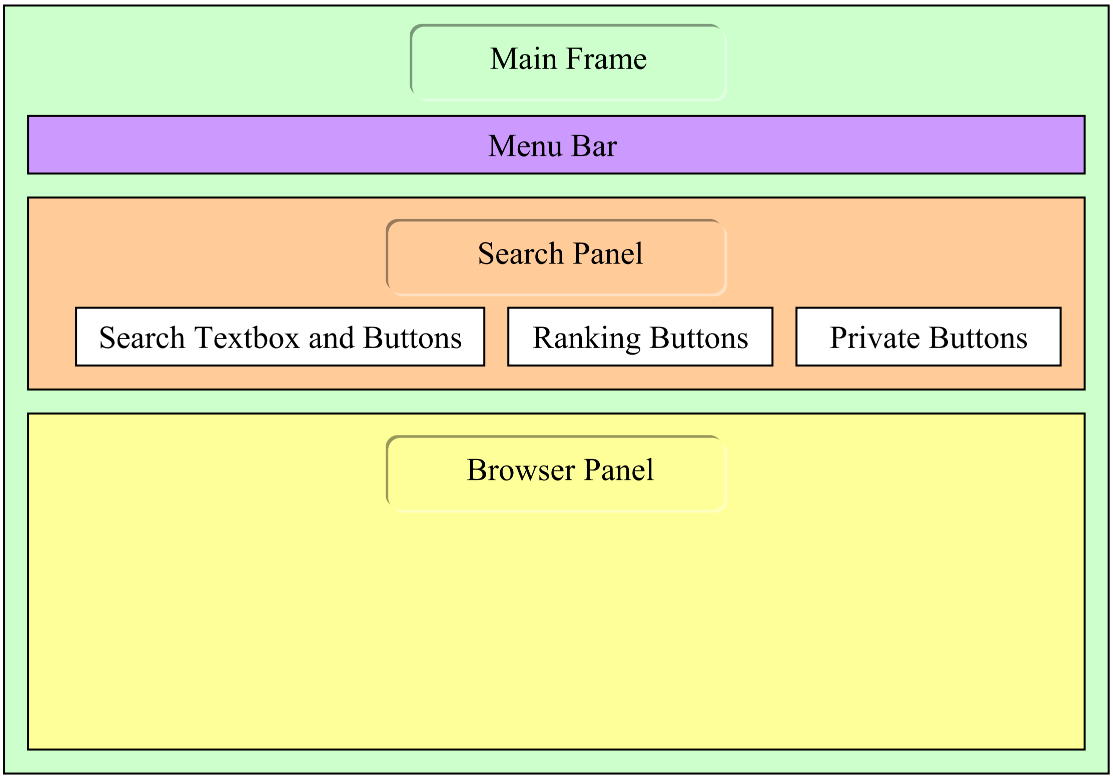
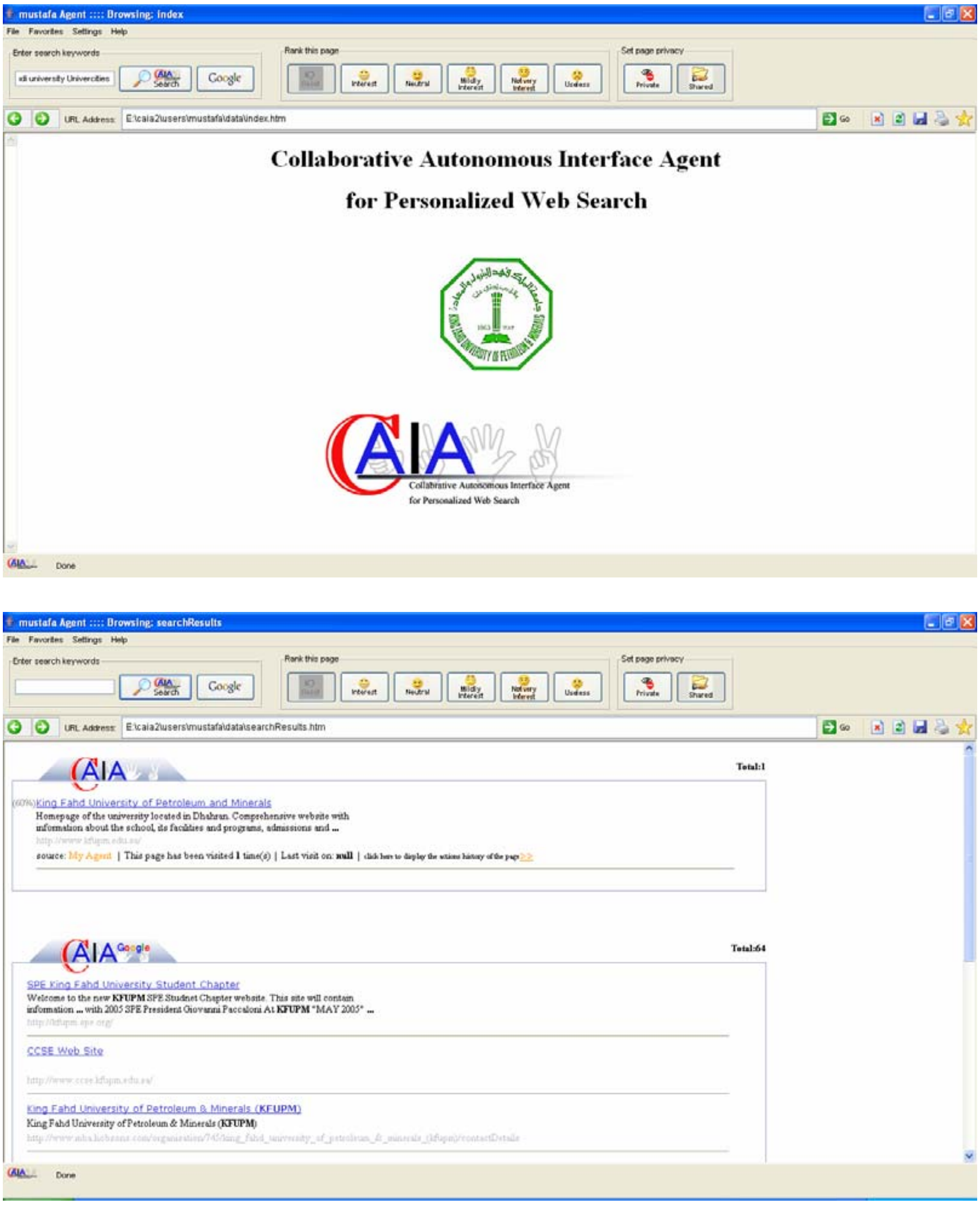


Figure 5.3 Design of the *Main Frame*

Main Frame has three main parts: *Menu Bar*, *Search Panel* and *Browser Panel*. *Menu Bar* has the menu items of the system such as *File-Menu-Item*. *Search Panel* has the search textbox, search buttons, the ranking buttons and the private buttons. Figure 5.4 illustrate snapshots of the main frame.



Enter search keywords: xli university Univercities

URL Address: E:\caia2\users\mustafa\data\index.htm

Collaborative Autonomous Interface Agent

for Personalized Web Search

URL Address: E:\caia2\users\mustafa\data\searchResults.htm

(60%)King Fahd University of Petroleum and Minerals
Homepage of the university located in Dhahran. Comprehensive website with information about the school, its facilities and programs, admissions and ...
source: My Agent | This page has been visited 1 time(s) | Last visit on: null | click here to display the actions history of the page>>

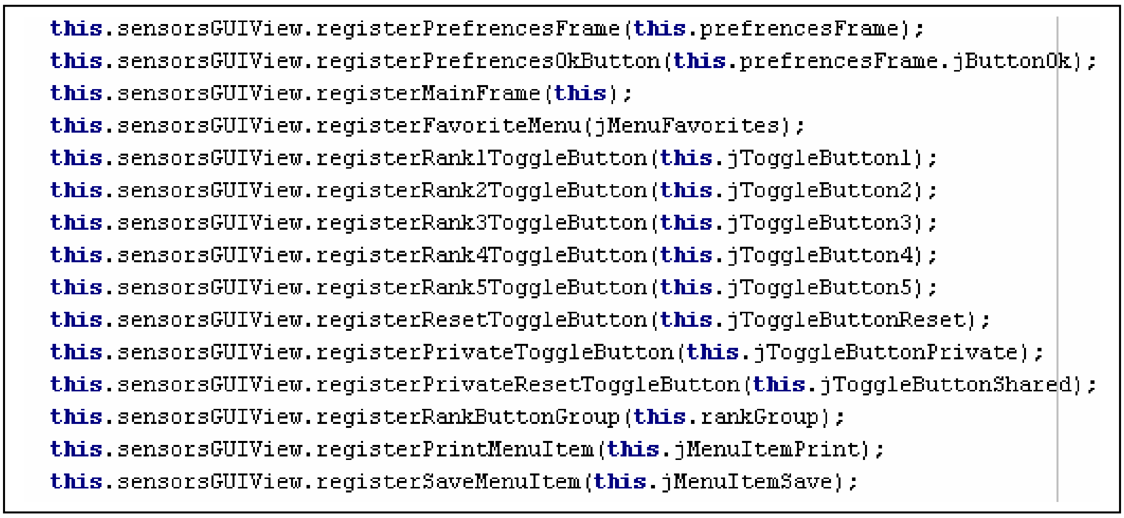
SPE King Fahd University Student Chapter
Welcome to the new **KFUPM** SPE Studnet Chapter website. This site will contain information ... with 2005 SPE President Giovanni Paccaloni At **KFUPM** "MAY 2005" ...

CCSE Web Site

King Fahd University of Petroleum & Minerals (KFUPM)
King Fahd University of Petroleum & Minerals (**KFUPM**)

Figure 5.4 Snapshots of the *Main Frame*

As mentioned earlier, *Main Frame* is responsible for registering all GUI components in the *Sensor Component* so that the *Sensor Component* listens to all GUI events. Figure 5.5 shows a sample code of the registration calls by the *Main Frame*.



```
this.sensorsGUIView.registerPrefrencesFrame(this.prefrencesFrame);
this.sensorsGUIView.registerPrefrencesOkButton(this.prefrencesFrame.jButtonOk);
this.sensorsGUIView.registerMainFrame(this);
this.sensorsGUIView.registerFavoriteMenu(jMenuFavorites);
this.sensorsGUIView.registerRank1ToggleButton(this.jToggleButton1);
this.sensorsGUIView.registerRank2ToggleButton(this.jToggleButton2);
this.sensorsGUIView.registerRank3ToggleButton(this.jToggleButton3);
this.sensorsGUIView.registerRank4ToggleButton(this.jToggleButton4);
this.sensorsGUIView.registerRank5ToggleButton(this.jToggleButton5);
this.sensorsGUIView.registerResetToggleButton(this.jToggleButtonReset);
this.sensorsGUIView.registerPrivateToggleButton(this.jToggleButtonPrivate);
this.sensorsGUIView.registerPrivateResetToggleButton(this.jToggleButtonShared);
this.sensorsGUIView.registerRankButtonGroup(this.rankGroup);
this.sensorsGUIView.registerPrintMenuItem(this.jMenuItemPrint);
this.sensorsGUIView.registerSaveMenuItem(this.jMenuItemSave);
```

Figure 5.5 Sample code of the registration calls by the *Main Frame*

Browser Panel

Browser Panel frame is responsible for displaying the search results page and the Website's content. It also provides browsing tool such as navigation and printing. *Browser Panel* consists of the following:

- *Status-Bar:* the status bar appears in the bottom of the page and displays the status of the browser such as "opening page http://xxx ".
- *Browser-Container*: we use ActiveX component to embed Internet Explorer using EZ JCom third party [34].
- *Browser-Tool-Bar:* consists of the navigation button, the website's address textbox and the options buttons such as printing and saving buttons.

Figure 5.6 illustrates *Browser Panel* design with all of its components.

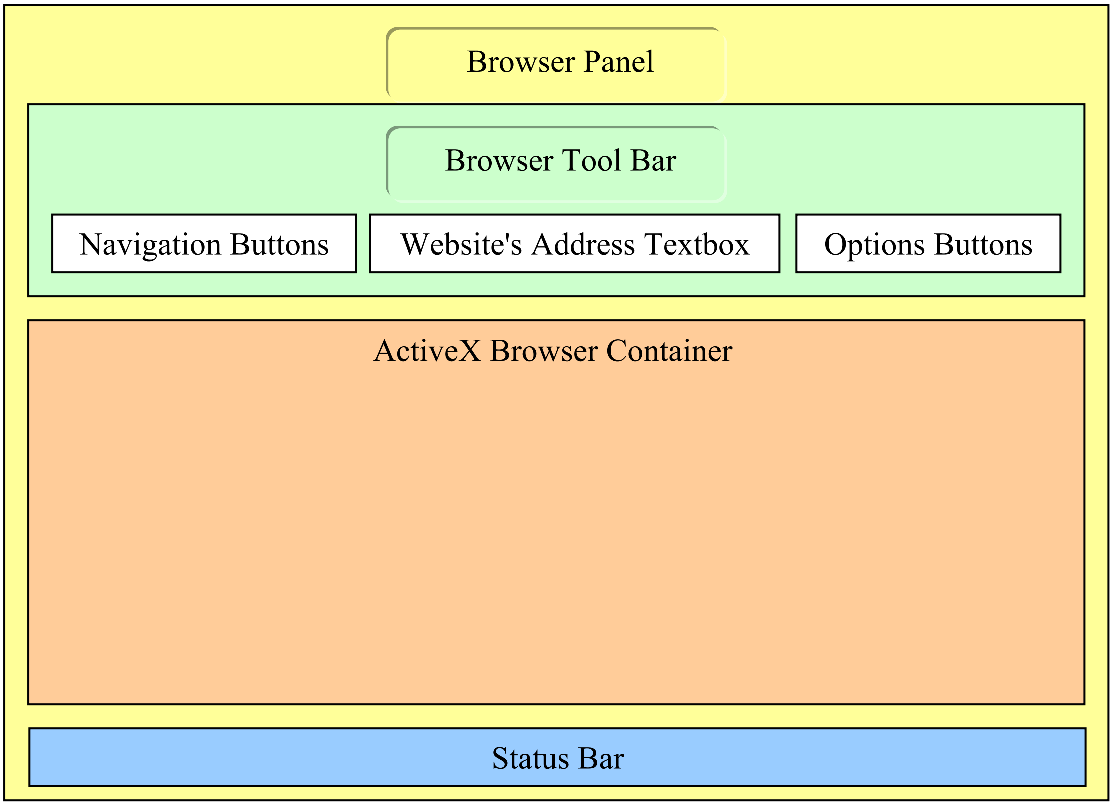
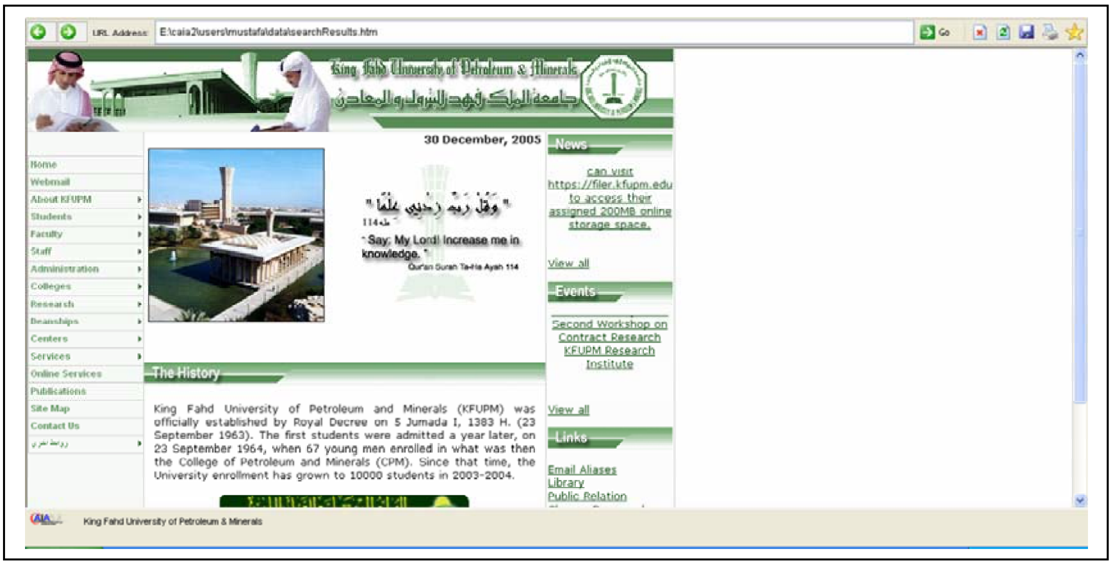


Figure 5.6 Design of the *Browser Panel Frame*

The following figure illustrates a snapshot of the *Browser Panel*.


URL Address: E:\caia2\users\mustafa\data\searchResults.htm

" وقل ربِّ زدني علمًا "
طه 114

" Say: My Lord! Increase me in knowledge. "
Qur'an Suran Ta-Ha Ayah 114

can visit https://filer.kfupm.edu to access their assigned 200MB online storage space.

Second Workshop on Contract Research KFUPM Research Institute

King Fahd University of Petroleum and Minerals (KFUPM) was officially established by Royal Decree on 5 Jumada I, 1383 H. (23 September 1963). The first students were admitted a year later, on 23 September 1964, when 67 young men enrolled in what was then the College of Petroleum and Minerals (CPM). Since that time, the University enrollment has grown to 10000 students in 2003-2004.

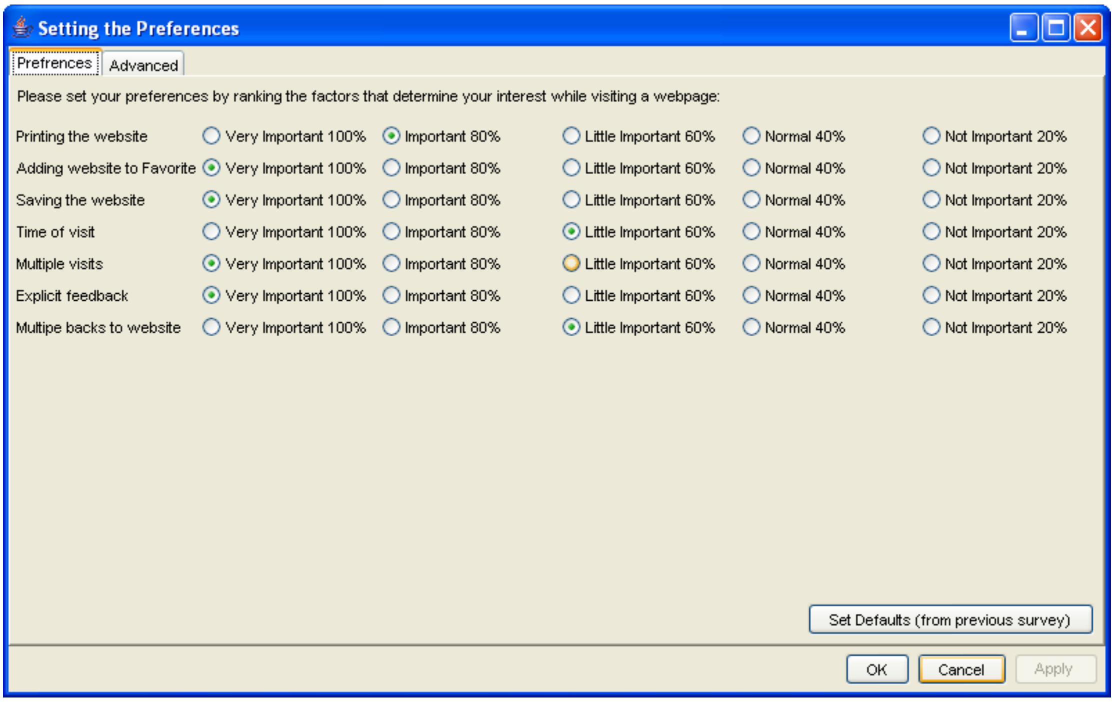
King Fahd University of Petroleum & Minerals

Figure 5.7 Snapshot of the *Browser Panel*

FramePreferences

Preferences Frame shows the options that allow the user to specify his/her preferences. The preferences are saved in a flat file called "Preferences.txt" in <app_dir>/users/<USER_NAME>/Prefreences.txt. The class of this frame is responsible to read and write on this text file. It is located in "Setting" menu in the main frame. The frame has two tabs:

a) *Preferences Tab*: it allows the user to set the weight of different factors that determine his/her interest in a Website. Here, the user can control his/her behaviors priorities. There is an option to set the default values based on a survey conducted previously and explained in Chapter-3. The following figure shows a snapshot of this tab.



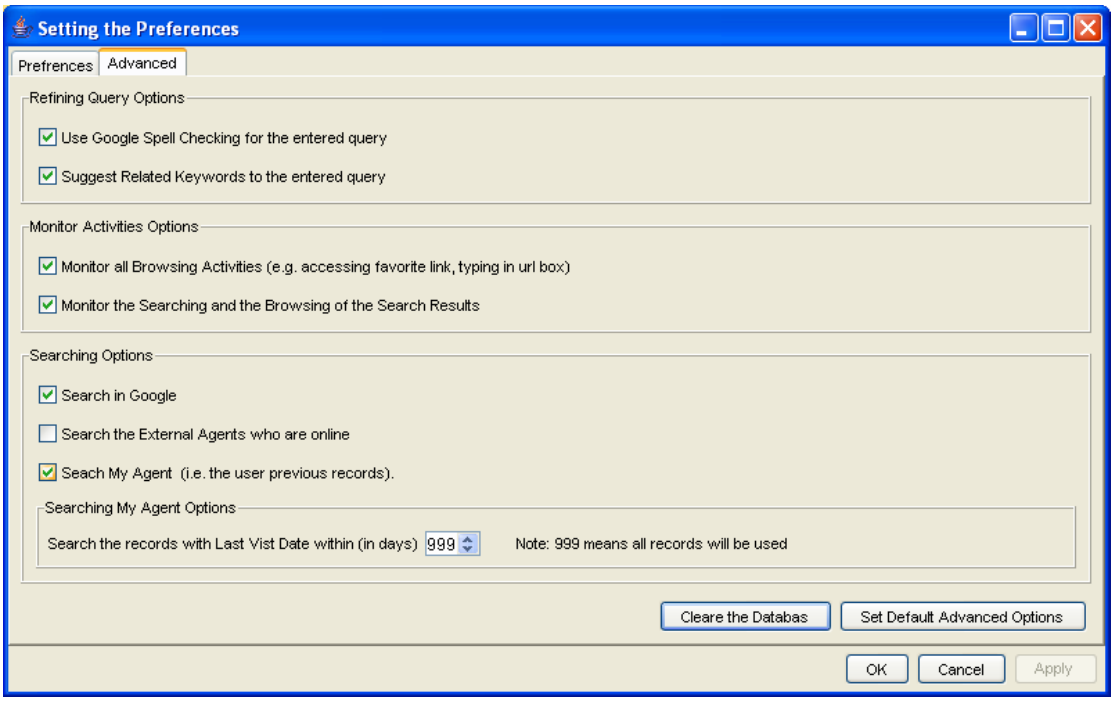
Please set your preferences by ranking the factors that determine your interest while visiting a webpage:

Printing the website	○ Very Important 100%	● Important 80%	○ Little Important 60%	○ Normal 40%	○ Not Important 20%
Adding website to Favorite	● Very Important 100%	○ Important 80%	○ Little Important 60%	○ Normal 40%	○ Not Important 20%
Saving the website	● Very Important 100%	○ Important 80%	○ Little Important 60%	○ Normal 40%	○ Not Important 20%
Time of visit	○ Very Important 100%	○ Important 80%	● Little Important 60%	○ Normal 40%	○ Not Important 20%
Multiple visits	● Very Important 100%	○ Important 80%	○ Little Important 60%	○ Normal 40%	○ Not Important 20%
Explicit feedback	● Very Important 100%	○ Important 80%	○ Little Important 60%	○ Normal 40%	○ Not Important 20%
Multipe backs to website	○ Very Important 100%	○ Important 80%	● Little Important 60%	○ Normal 40%	○ Not Important 20%

Figure 5.8 Snapshot of the *Preferences-Tab* in the *Preferences-Frame*

b) *Advanced Tab*: it gives the user the flexibility to set his/her preferences as explained below followed by a figure that shows a snapshot of this tab.

- *Refining Query Options*:
    i. If the user wants the system to use Google spell-checker to spell-check his/her query.
    ii. If the user wants the system to suggest related keywords to the user's query based on his/her *User Profile*.
- *Monitor Activities Options*:
    i. If the user wants the system to monitor all his/her all browsing activities, i.e. when s/he accesses any Website directly without searching for them using the Website's address textbox.
    ii. If the user wants the system to monitor all his/her behaviors on the searching results after s/he searches through the system.
- *Searching Options:*
    i. If the user wants the system to search in Google using the Google APIs with the refined keywords.
    ii. If the user wants the system to search in other external agents in the community.
    iii. If the user wants the system to search in his/her agent based on his/her *User Profile*. Also, s/he can set the duration to search within, i.e. s/he can ignore the past history with this option.
    iv. If the user wants the system to clear his/her *User Profile* (database).
    v. If the user wants to set all above options to the default values.



Refining Query Options

☑ Use Google Spell Checking for the entered query

☑ Suggest Related Keywords to the entered query

Monitor Activities Options

☑ Monitor all Browsing Activities (e.g. accessing favorite link, typing in url box)

☑ Monitor the Searching and the Browsing of the Search Results

Searching Options

☑ Search in Google

☐ Search the External Agents who are online

☑ Seach My Agent (i.e. the user previous records).

Searching My Agent Options

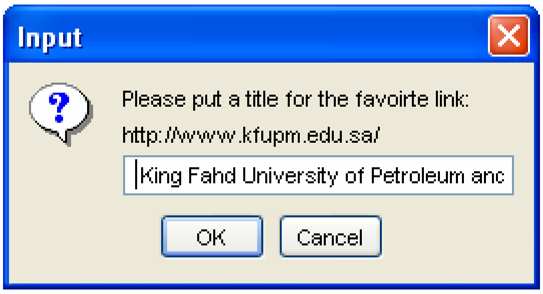
Search the records with Last Vist Date within (in days) 999

Note: 999 means all records will be used

Figure 5.9 Snapshot of the *Advanced-Tab* in the *Preferences-Frame*

Favorite

This class is responsible to store the favorite links. The main frame serialized the object of this class for saving the attributes and de-serializing for reading the data. It shows a dialog that allows user to enter the title of the favorite link. Each link is represents by an instance of "*FavoriteLink*" bean. *Favorite* class has an array of "*FavoriteLink*". The following figure shows a snapshot of the dialog appears to the user when s/he asks to add the current displayed Website to his/her favorites.



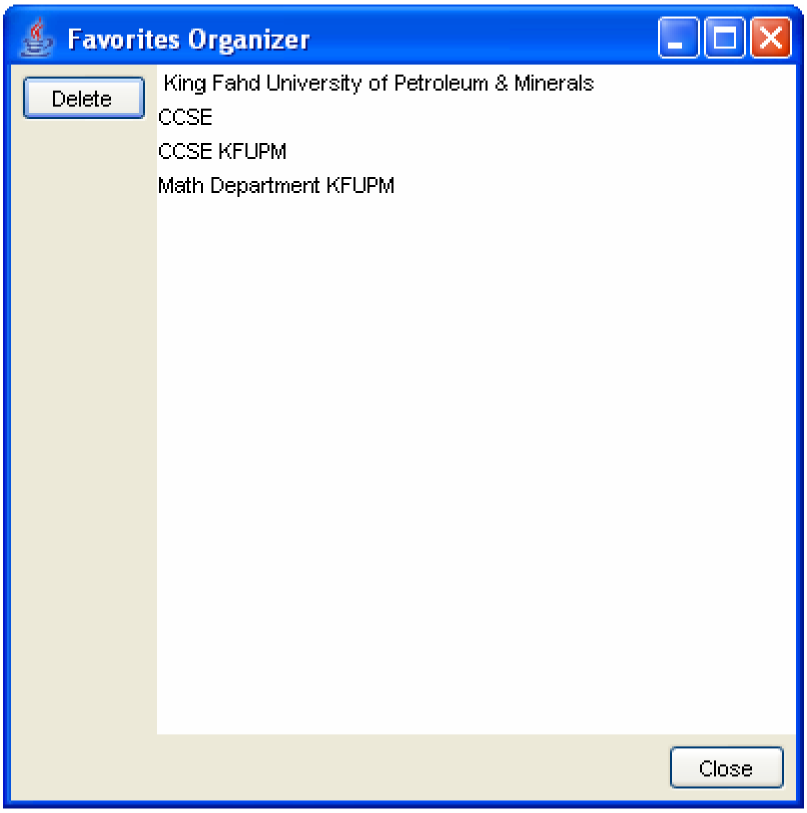
Please put a title for the favoirte link:
http://www.kfupm.edu.sa/

King Fahd University of Petroleum anc

Figure 5.10 Snapshot of the favorite addition dialog

FrameOrganizeFavorites

This frame shows all links in a list box. It gives the user the ability to delete unwanted favorites. The following figure shows a snapshot of this frame.

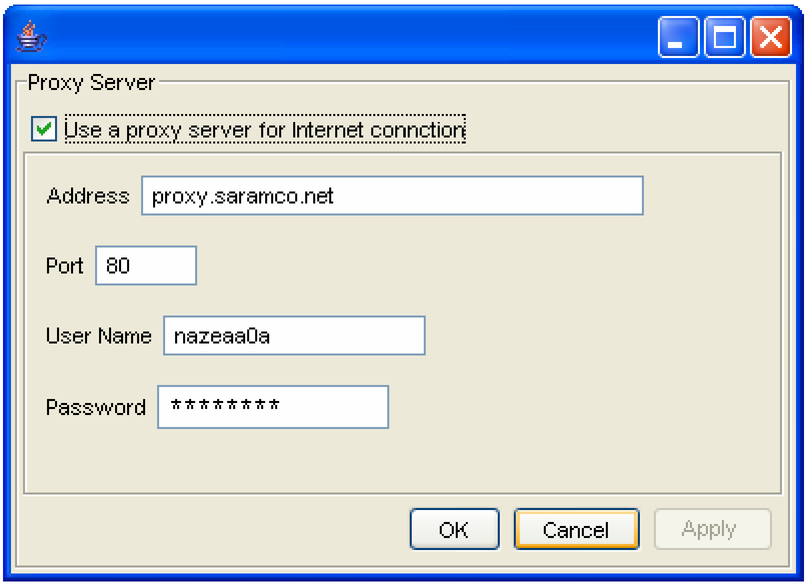


King Fahd University of Petroleum & Minerals
CCSE
CCSE KFUPM
Math Department KFUPM

Figure 5.11 Snapshot of the *Favorites-Organizer-Frame*

FrameProxy

It gives the user the option of using proxy. It allows him/her to enter the proxy address, port, user name and password. The proxy settings are saved in a flat file called "ProxySettings.txt" in <app_dir>/users/<USER_NAME>/ProxySettings.txt after encrypting the password. Proxy settings menu item is located in "Setting" menu in the main frame. The following figure shows a snapshot of this frame.



Proxy Server

☑ Use a proxy server for Internet connction

Address proxy.saramco.net

Port 80

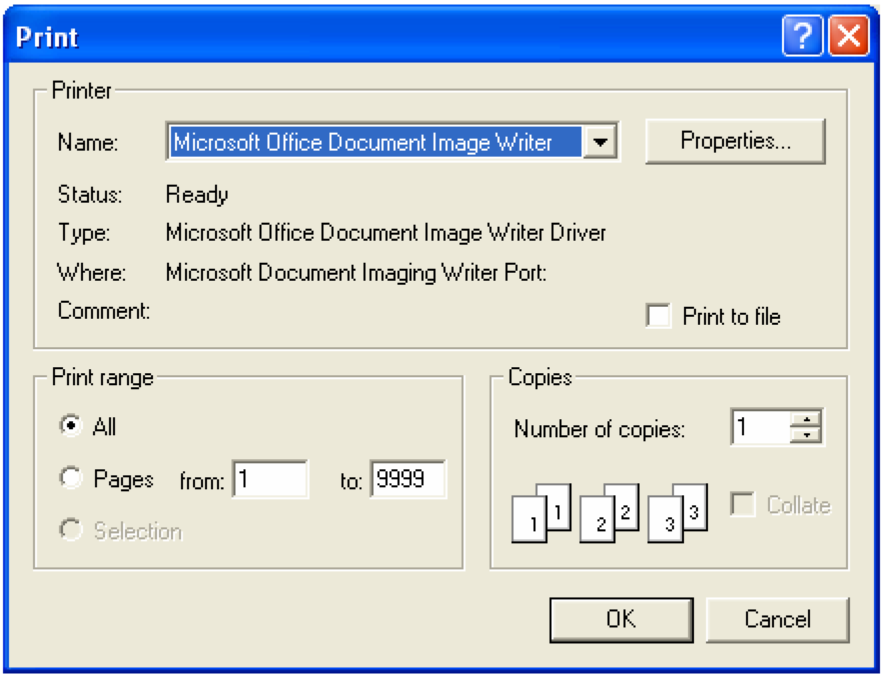
User Name nazeaa0a

Password *********

Figure 5.12 Snapshot of the proxy's settings frame

PrintUtilities

PrintUtilities displays a print dialog for printing the currently browsed Website. The following figure shows a snapshot of the print dialog.



Printer

Name: Microsoft Office Document Image Writer

Status: Ready

Type: Microsoft Office Document Image Writer Driver

Where: Microsoft Document Imaging Writer Port:

Comment:

☐ Print to file

Print range

● All

○ Pages from: 1 to: 9999

○ Selection

Copies

Number of copies: 1

☐ Collate

Figure 5.13 Snapshot of the print dialog

II. THE SENSOR COMPONENT

Sensor Component is responsible for listening to all events and actions of the registered items of the *GUI Component* and then passing the actions to the *Brain Component*. The *GUI Component* has the option to let its items to be listened and monitored by the *Sensor Component*. *Sensor Component* is implemented in a separate package called *sensors*. The implementation corresponds to the design described in Chapter-4. Table-5.2 summarizes Java classes of the *sensors* package. The more detailed explanation of important classes follows in the next subsections. Each class is saved in a file of the same name with the extension ".java".

Class name	Task of the class
Event	Bean class of the events.
Preferences	Bean class of the preferences settings.
Proxy	Bean class of the proxy settings.
SensorGUIView	Class deals with the interactions between the *Sensor Component* with the *GUI Component*.
URL	Bean class of the URLs.

Table 5.2 Classes of the *Sensor Component*

Sensors GUI View

It is a singleton and huge because it acts as a mediator of the GUI items. It holds all registration methods that *GUI Components* use to register its items to be listened into the *Sensor Component*. It has a list of internal action listeners for each GUI-item.

URL

This is a bean class that holds the URL information illustrated in Table 5.4. Moreover, it has other attributes like the explicit rate of the URL. Table 5.3 illustrates those attributes for the explicit rating. They are static and centralized in this class so they can be changed only here and the change will be reflected in all other classes use the explicit rate attributes.

Attribute	Type	Details
RANK1	String	Static string holds a value of "Interest" used in the explicit rate.
RANK2	String	Static string holds a value of "Neutral" used in the explicit rate.
RANK3	String	Static string holds a value of "Mildly interest" used in the explicit rate.
RANK4	String	Static string holds a value of "Not very interest" used in the explicit rate.
RANK5	String	Static string holds a value of "Useless" used in the explicit rate.

Table 5.3 *URL*'s static attributes for explicit rating

Next table summarizes the bean attributes of the *URL* class. There are two methods: *setter()* and *getter()* for each attribute. These methods are responsible for setting and getting the attribute's value.

Attribute	Type	Details
url	String	The address of the URL.
Title	String	The title of the URL.
Events	ArrayList	List of "Event" objects representing the events occurred to the URL
Rank	String	Explicit rate of the URL.
urlKeywords	ArrayList	Keyword of the URL obtained from the title and the summary.
userKeywords	ArrayList	Keywords of the URL obtained from the user when s/he searches in the system.
deletedKeywordsByTheUser	ArrayList	List of the keywords deleted by the user after it is suggested by the system.
searchedUrl	Boolean	Determines if the URL comes from a search process.
addedToDBInCurrentSession	Boolean	Determines if the URL is already added to the database in the current session.
Summary		The summary of the URL.
privateUrl	Boolean	Determines if the URL is private.
noOfVisits	Integer	Number of visits of the URL.
lastVisit	String	Date and time of the last visit of the URL.
weightsForAllKeywords	String	Total weight for all keywords of the URL.

Table 5.4 *URL* bean attributes

Event

This class is used to standardize the event handling in the system. It is a bean that holds the name of the event and the date of it. It has also static strings that hold the event types as follows:

- *START_VISIT:* an event issued when the user starts visiting a Website.
- *END_VISIT:* an event issued when the user ends visiting a Website.
- *SAVED:* an event issued when the user saves the visited Website.
- *ADDED_TO_FAVORIATE:* an event issued when the user adds the visited Website to his/her favorites.
- *PRINTED:* an event issued when the user prints the visited Website.

Proxy

This class is used to standardize the proxy settings handling in the system. It is a bean that holds whether the user wants to use a proxy or not. It holds also the proxy address, proxy port, user name and password. All the attributes are settled by the GUI Component after it gets the proxy settings from the user through the Proxy Frame. More details about proxy frame are in Section I in this chapter.

Preferences

This class is used to standardize the *Preferences* settings handling in the system. It is a list of settings of the preferences of the user. All the attributes are settled by the GUI Component after it gets the *Preferences* settings from the user through the *Preferences* Frame. More details about *Preferences* frame are in Section I in this chapter.

III. THE BRAIN COMPONENT

Brain Component is the core component of the system. It is responsible for refining the user's query and then searching for it in the User Profile. It is also responsible for monitoring the user's behaviors and reflecting them in his/her User Profile. *Sensor Component* as explained in the previous section is carrying the user's behaviors into the *Brain Component*.

Brain Component has two windows, one to the *Sensor Component* and the other to the *GUI Component*. These two windows are implemented in the classes: *Brain-GUI-View* and *Brain-Sensor-View*.

Brain Component has a lot of artificial intelligent algorithms. The core algorithms are implemented in the classes: *Monitor Events* and *Search Processing*. *Monitor Events* has the algorithms that manipulate the weights of the keywords based on the user's behaviors. *Search Processing* has the algorithms that refine and search the user's query.

Brain Component has two main bean classes that are used when retrieving the search results whether from the user's agent, other agents or Google search engine. The first one is *Search-Result* which represents the whole results of a search quarry and holds a list of "*Search-Result-Element*" objects. The second one is *Search-Result-Element* which represents a single URL found in the search process of the user's query.

Brain Component is implemented in a separate package called *brain*. The implementation corresponds to the design described in Chapter-4. Table-5.5 summarizes Java classes of the *brain* package. The more detailed explanation of important classes follows in the next subsections. Each class is saved in a file of the same name with the extension ".java".

Class name	Task of the class
BrainController	*Brain Component*'s classes controller.
BrainGUIView	Class deals with the interactions between the *Brain Component* and the *GUI Component*.
BrainSensorView	Class deals with the interactions between the *Brain Component* and the *Sensor Component*.
CAIAAgent	The main agent's class in the system. It handles the collaboration between agents. More detail is in Section IV.
GoogleAPI	Class handles the communication with Google spell-check and Google search engine thorough the Google APIs.
Keyword	Bean class holds the keyword's information such as weight.
MonitorEvents	Class handles the user's behaviors in the browser and updates the User Profile.
SearchProcessing	Class handles the refine and search processes.
SearchResult	Bean class holds the whole results of a query's search. It has a list of "*SearchResultElement*" objects.
SearchResultElement	Bean class holds a single URL found in the search process. It has three attributes: the source of the element (search engine, the user's agent or another agent), search keywords and *URL* object that's created in the *Sensor component*.
Util	Static class holds utilities methods.
WordCounter	Bean class holds a word and its count used by *Util* class.

Table 5.5 Classes of the *Brain Component*

BrainController

This class is a singleton class that is available all the time to control and serve other entities in the component. It has other functionalities described in the following.

i. It is initializing and creating instances of the following objects:
    a. *MonitorEvents* object which takes care of the user's behaviors.
    b. *SearchProcessing* object which takes care of the refinement and search.
    c. *BrainGUIView* object which interfaces with *GUI Component*.

ii. It is responsible for the connection with the database and any operation done in the database. It has the following methods which serve its purpose:
    a. *deleteLowWeightedKeywords():* this method is called once the agent is up to clear the keywords in the User Profile which has low weight i.e. it has a weigh less than 0.2.
    b. *clearDatabase():* this method is called when the user clicks on "*Clear Database*" button in the *GUI Component*. It then clears the User Profile which contains all the visited Websites and their keywords.
    c. *connect()*: this method is responsible for connecting to the database. It is called once the agent starts.
    d. *execute()*: this method is executing queries. It is called from different classes in the *Brain Component* to execute different queries. It gets a SQL statement as input and returns a *recordset* contains the results of the executing of the query.
    e. *close()*: this methods closes the connection with the database. It is called once the system exit.

- f. *isKeywordExist():* this method determines if a certain keyword exist in the User Profile (the database).

iii. It is responsible for the following beans:

- a. *rs:* it represents the opened *recordset*. Its type is *Recordset* object. It has both setter and getter methods to set and get its value.
- b. userKeywords: it represents the user's query as *ArrayList* object contains all the entered keywords. It has a getter method which gets the value of this *ArrayList*. It has also a setter method which do three tasks:
    - i. setting the user keywords in the *Search-Processing* object,
    - ii. calling the *search()* method in the *Search-Processing* object which is responsible for the refinement and search processes,
    - iii. and then setting the *Search-Result* object in the *BrainGUIView* after getting the *search()* method completes its task.
- c. *proxySettings:* it represents the proxy's settings. Its type is *ProxySettings* object. It has both setter and getter methods to set and get its value.
- d. *filesSettings:* it represents the files' settings. Its type is *FileSettings* object. It has both setter and getter methods to set and get its value.
- e. *myAgent*: it represents the user's agent. Its type is *CAIAAgent* object. It has both setter and getter methods to set and get its value.
- f. *preferencesSettings:* it represents the preferences' settings. Its type is *PreferencesSettings* object. It has both setter and getter methods to set and get its value.

g. *searchResult:* it represents the opened *recordset*. Its type is *SearchResult* object. It has both setter and getter methods to set and get its value.
h. *urls:* it represents the currently visited URLs. Its type is *ArrayList* that contains a list of *URL* objects. It has a getter method which gets the value of this *ArrayList*. It has also a setter method which sets its value and call the method *monitorUrls()* in *MonitorEvents* object which is responsible for monitoring the user's behaviors and reflecting them in the User Profile.

BrainGUIView

This class is responsible for any interaction between the *Brain Component* and the *GUI Component*. It sends the search result to the *GUI Component* after performing a search process in order to display them in the *Browser-Panel-Frame*. It is also responsible to get confirmed user's query after the Brain Component suggests refined keywords to the user. So it sends the refined keywords to the *GUI Component* and gets the keywords the user interested in.

BrainSensorView

This class is responsible for any interaction between the *Brain Component* and the *Sensor Component*. It gets the following from the *Sensor Component* and then sets them in the brain controller: user's query to be searched, user's preferences settings, and user's proxy settings.

BrainSensorView also gets from the *Sensor Component* the URLs that has been visited with all the events performed on that URL.

GoogleAPI

This class is responsible for interfacing with Google spell-check and Google search engine in the Internet. It sends the user's query and gets the correct spelled keyword. Also, it sends a refined query and gets the search results from Google search engine.

MonitorEvents

The main functionality of this class is taking care of user's behaviors on the visited Websites. *Sensor Component* sets the visited URLs and all the events occurred in the *Brain Controller* which calls the main method in this class, *monitorUrls()*. This method has the following algorithm:

```
(For each URL in the visited URLs) {
    Loop on the events of the URL {
        Adding weight to the URL's keywords based on the event's type.
    }
    Checking the privacy and the explicit ranking of the URL
    Adding the URL with to the User Profile if it is not exist
    Adding the events done to the URL to the User Profile
    Adding the URL's keywords to the User Profile and associate them to the URL
    }
```

Figure 5.14 Algorithm for the method *monitorURL()*

Adding weights for the keywords is based on the event type. The following table gives the weight for each type as these values are the default values based on the survey conducted previously; see Chapter-3 for more details.

Event type	Weight
Entered keyword by the user	0.5
Removed keyword by the user (from the suggested refined keywords)	-0.5
Explicit rate: "Interest"	1.0 * 0.1
Explicit rate: "Neutral"	0.7 * 0.1
Explicit rate: "Mildly interest"	0.0 * 0.1
Explicit rate: "Not very interest"	-0.7 * 0.1
Explicit rate: "Useless"	-1.0 * 0.1
Print the Website	0.2
Add the Website to favorite	0.5
Save the Website	0.3
Time of Visit	0.4
Multiple visits to the same Website	0.5
Multiple backs	0.5

Table 5.6 Weights for different event types

SearchProcessing

The main functionality of this class is taking care of the refinement and search processes after the user enters a query and initiates a search request to that query. When *Sensor Component* senses that the user enters a new query and asks the system to search for it, it sends it to the *Brain Component*. It arrives to the *Brain Controller* which then calls the main method in this class, *Search()*. This method has the following algorithm:

```
Create a new object of SearchResult and assign it a random search ID.
Filter the user's query by removing the noise words.
Spell-check the user's query
Refine the user's query by adding relevant keywords from his/her User Profile
Send the suggested query to the GUI Component to get the user's confirmation
Start searching the user's agent and add any found URL to the SearchResult object
Start searching the other agents and add any found URL to the SearchResult object
Send the refined query to Google and add any found URL to the SearchResult object
```

Figure 5.15 Algorithm for the method *Search()*

Util

This class is static class and has static utilities methods which are like services. These methods can be used without the need of creating a new object of this class. It has the following methods:

- *arrayList2String*(): this method is converting *ArrayList* object into *String* object. It takes an *ArrayList* object as input and returns a *String* object as an output. It loops though all objects in the *ArrayList* and adds them to the output *String* object.

- *string2ArrayList():* this method is converting *String* object into *ArrayList* object. It takes a *String* object as input and returns an *ArrayList* object as an output. It uses *StringTokenizer* object in order to tokenize the input string into an array. Then, it reads all String objects in the array and adds them to the output *ArrayList* object.
- *removeDuplicatedWords():* this method is removing duplicated words from an *ArrayList* object. It takes an *ArrayList* object as input and returns an *ArrayList* object as an output. It checks the input ArrayList object item by item and remove not only the duplicated word but also it removes the word that is found in two formats single and not single. For example, if it finds the words: suggestion and suggestions, it will remove one of them. It takes care also the different cases, capital and small letters. The algorithm used is as following: it takes the words of the input *ArrayList* object word by words and adds them to the output *ArrayList* object provided that it is not added yet.
- *isDuplicatedWordExist():* this method is checking if an input word exists in an input *ArrayList* object. It takes a String object representing the input word and an *ArrayList* object as input and returns a Boolean output. It checks the input word against all words in the input ArrayList object. It then returns the Boolean output *true* if the word exist in the *ArrayList* object otherwise it returns *false*.
- *isNumberExistInArrayList():* this method is checking if an input number exists in an input *ArrayList* object. It takes an Integer object representing the input number and an *ArrayList* object as input and

returns a Boolean output. It checks the input number against all numbers in the input ArrayList object. It then returns the Boolean output *true* if the number exist in the *ArrayList* object otherwise it returns *false*.

- *time():* this method is returning the current time as a String. It reads the current time using the Java class: " *java.util.Date()*" and then coverts it to a string understood by all databases as date-time format.
- *getKeyowrdFromURL():* this method is providing the keywords from the URL's address. It takes a String object representing the URL's address and returns an ArrayList object contains all the keywords in it.
- *getMostFrequentWordInArrayList():* this method is finding the most frequent word in an *ArrayList* object. It takes an *ArrayList* object that contains a list of words and returns the most frequent word as *String* object. It counts the number of occurrences of each word in the input *ArrayList* object and then returns the word with the highest count.
- *getDeletedWords():* this method is returning the deleted words in an *ArrayList* object. It takes an *ArrayList* object representing the original array and anther *ArrayList* object representing the changed array as input. It then checks the words missing in the changed array and returns an *ArrayList* object contains all the deleted words as an output.

IV. THE COMMUNICATION BETWEEN AGENTS

CAIAAgent class in *Brain Component* is the main agent's class in the system. It is the staring point of the system where an agent is created and registered in *JADE DF* directory in order to collaborate with other agent. The DF (Directory Facilitator) provides a Yellow Pages service by means of which an agent can find other agents providing the services he requires in order to achieve his goals. We used the Yellow Pages service provided by the DF agent. It handles the collaboration between the user's agent and other agents.

We created simple *JADE* agents and made them executing tasks and communicate between each other. *JADE* is a middleware that facilitates the development of multi-agent systems. It includes a runtime environment where JADE agents can “live” and that must be active on a given host before one or more agents can be executed on that host.

In a collaborative agent model, a particular agent may not have any prior knowledge, while there may be a number of agents belonging to other customers who do. We used JADE model to make the agents collaborate with each others. The following Figure gives an idea where *JADE* is a middle layer to get agents talk with each other.

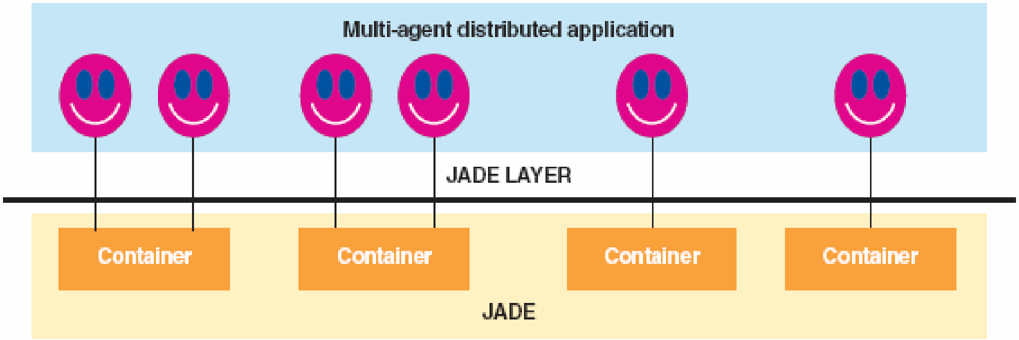


Figure 5.16 *JADE* layer

We have used the following packages in *JADE*:

- *jade.core* which implements the kernel of the system. It includes the *Agent* class that was extended in our system's main class, *CAIAAgent*.
- *Behavior* class which is contained in *jade.core.behaviours* sub-package. *Behaviors* implement the tasks, or intentions, of the agent. They are logical activity units that can be composed in various ways to achieve complex execution patterns and that can be concurrently executed.
- *jade.lang.acl* sub-package which is provided to process *Agent Communication Language* according to *FIPA* standard specifications. For more information regarding *FIPA* specification, refer to [36].
- *jade.domain* package which contains all classes that represent the Agent Management entities defined by the FIPA standard, in particular the *AMS* and *DF* agents, that provide life-cycle, white and yellow page services.

The Agent Management System (AMS) is the agent who exerts supervisory control over access to and use of the Agent Platform. Only one AMS will exist in a single platform. The AMS provides white-page and life-cycle service, maintaining a directory of agent identifiers (AID) and agent state. Each agent must register with an AMS in order to get a valid AID. The Directory Facilitator (DF) is the agent who provides the default yellow page service in the platform. The Message Transport System, also called Agent Communication Channel (ACC), is the software component controlling all the exchange of messages within the platform, including messages to/from remote platforms. The following figure illustrates the FIPA architecture of the Agent platform.

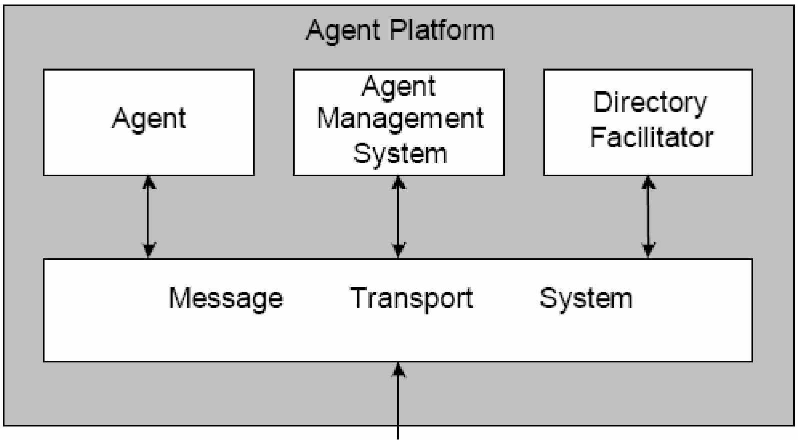


Figure 5.17 *FIPA* specification of Agent platform

We defined the agent operations by writing the following behaviors in *CAIAAgent*:

- *RequestAgent*: it extends *Behavior* class and it is responsible for sending the search request from the user's agent to other live agents. The *search()* method in *SearchProcessing* object is creating a new instance of this class once it starts searching in external agents.
- *ReplyAgent*: it extends *CyclicBehaviour* class which is a child of *Behavior* class. The special characteristic in *CyclicBehaviour* object is that it is executing forever and hence listening to any event while the normal Behavior object is executed once requested only. We implemented *ReplyAgent* to provide a search service for other agents. Once the user's agent gets a search request, it calls the method *searchForExternalAgents()* which is located in the *SearchProcessing* object. This method is returns the search result as String.

CHAPTER 6

ESTING AND EVALUATION

In this chapter we demonstrate a sample scenario that shows the benefits of CAIA in terms of improving the user's query and decreasing the number of retrieved results and hence increasing the precision. We have several experiments to evaluate CAIA's effectiveness as effectiveness measures both:

- The ability of CAIA to satisfy the user in getting the relevance URLs
- The ability of CAIA to learn and adapt User's Profile over time

User's Profile (UP) contains all URLs visited by the user and their relevant keywords with a weight for each keyword. Weights are assigned and adapted based on the survey as explained in Chapter 3.

I. SAMPLE SCENARIO

Assume the user is using CAIA; he is a none-Saudi and knows nothing about the universities in Saudi Arabia. In the first usage of CAIA, his interest is to find out a good university at KSA. Let's assume he got interested in KFUPM. In the second search, he is looking to find a good collage in KFUPM. The third search is to look at the department of Computer Science at KFUPM. CAIA should help him in personalizing his search. The result was as follows:

- Quarry 1: the user intention is to find out a good university in KSA. We assume the user got interested in KFUPM. So, the target of this quarry is KFUPM Website.

User Input	*Saudi university*
Suggested Key words	*N/A*
No. of results	
My CAIA=	*N/A*
CAIA Google=	*1170,000*
External Agent=	*N/A*
Track	*Hits the second link in the first page.*
Actions	*Add in Favorite + rank interest*

Table 6.1 Sample scenario for Query-1

- Quarry 2: the user intention now is to search in a good collage in KFUPM. He is using the same keyword because he is expecting CAIA will understand his interest.

User Input	*Saudi university*
Suggested Key words	*Saudi university Mineral Petroleum King Fahd*
No. of results	
My CAIA=	*1*
CAIA Google =	*29*
External Agent=	*N/A*
Track	*Hits the first link in the search result + hit an internal link in KFUPM Website to enter to CCSE Website*
Actions	*Add Favorite + Rank interest*

Table 6.2 Sample scenario for Query-2

- Quarry 3: at this time, the user is looking to visit the ICS Website. He is expecting CAIA will understand that he is involved in CCSE at KFUPM

User Input	*Saudi university*
Suggested Key words	*KFUPM CCSE Saudi university*
No. of results	
My CAIA=	*2*
CAIA Google =	*20*
External Agent=	*N/A*
Track	*Hits the first link in the search result + hit an internal link in CCSE Website to enter to ICS Website*
Actions	*Add Favorite + Rank interest*

Table 6.3 Sample scenario for Query-3

- Quarry 4 (collaboration): Now, he is looking to visit the Math department Website. He knows that his friend is studying in that department. He will write only "math" as a user input because he is expecting that his CAIA is smart enough and will understand that he means the math department at KFUPM.

User Input	*Math*
Suggested Key words	*Math*
No. of results	
My CAIA=	*0*
CAIA Google =	*221,000,000*
External Agent=	*1*
Track	*Hits the first link in the first page.*
Actions	*Add Favorite + Rank interest*

Table 6.4 Sample scenario for Query-4

II. THE ANALYSIS OF THE PRECISION

Precision is how much results related to the user, i.e. results that meet the user needs. It has a well-known equation as following:

$$\text{Precision} = \frac{\text{The number of relevant URLs retrieved}}{\text{Total number of URLs retreived}} \qquad \text{Equation (2)}$$

Precision is calculated by dividing the number of relevant URLs retrieved on the total number of URLs retrieved. Precision tells us how much improvements are the query expansion and filtration of the retrieved result. In this experiment, we submit twenty queries to Google and then submit the same queries to the CAIA with its advantages of enhancing the queries and filtering the results. The results depicted in the next table and figure show the facts the CAIA is helpful for the users in terms of retrieving relevant information consistent with the users' information need.

We assume the number of relevant URLs is constant and we observed the number of Google results and CAIA results. So, we calculate the precision based on the total number of retrieved URLs and assume there are only twenty relevant URLs.

Query	Number of Results		Precision	
	CAIA	Google	CAIA	Google
Query1	26	11600	76.92	0.17
Query2	27	12200	74.07	0.16
Query3	77	34100	25.97	0.06
Query4	68	30040	29.41	0.07
Query5	25	11250	80.00	0.18
Query6	23	10050	86.96	0.20
Query7	45	20050	44.44	0.10
Query8	28	12300	71.43	0.16
Query9	34	15000	58.82	0.13
Query10	28	12550	71.43	0.16
Query11	51	22520	39.22	0.09
Query12	34	15230	58.82	0.13
Query13	33	14520	60.61	0.14
Query14	33	14520	60.61	0.14
Query15	23	10050	86.96	0.20
Query16	42	18620	47.62	0.11
Query17	42	18520	47.62	0.11
Query18	59	26380	33.90	0.08
Query19	64	28600	31.25	0.07
Query20	28	12500	71.43	0.16

Table 6.5 Number of results and precision for both CAIA and Google

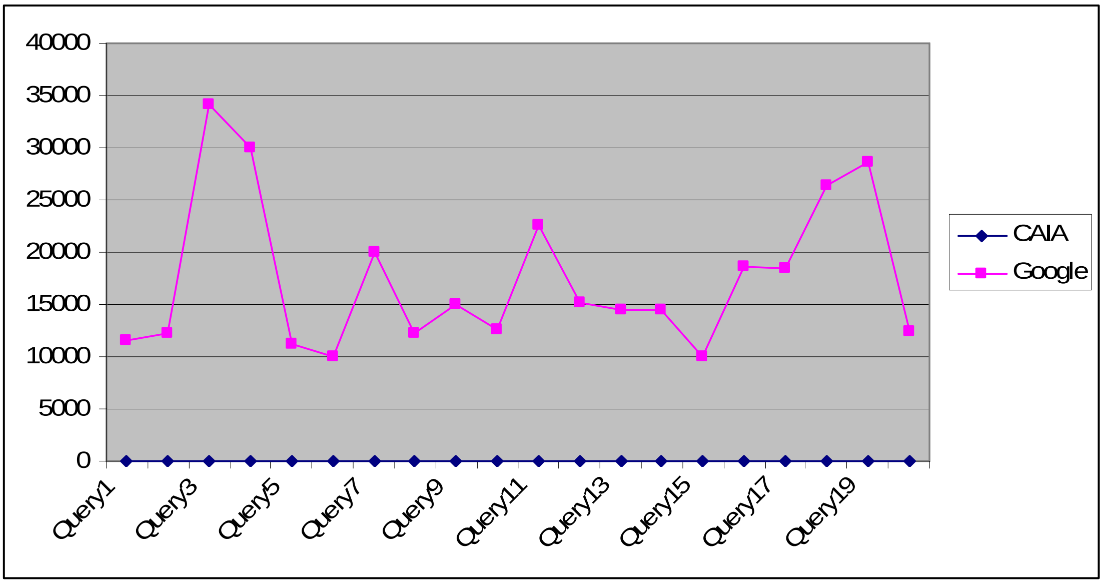


Figure 6.1 Chart of the comparison of number of results between CAIA and Google

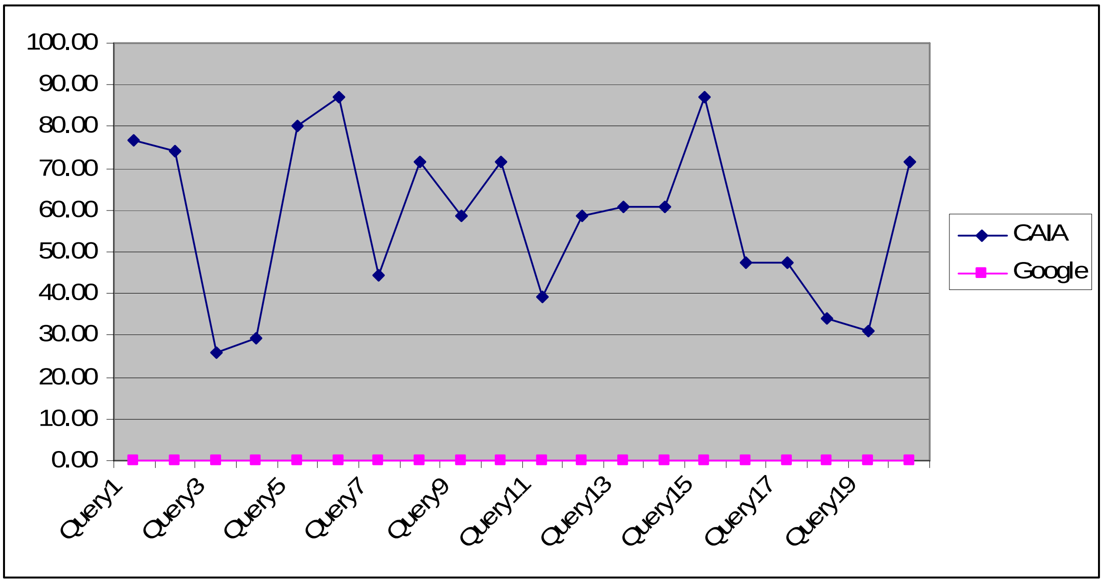


Figure 6.2 Chart of the comparison of precision between CAIA and Google

Also, CAIA is improving over trials by decreasing number of results and hence increasing the precision. Next table and figures shows the experiment of twenty queries with three trials in CAIA and the improvement is clear.

	Number of Results			Precision		
Query	Trial1	Trial2	Trial3	Trial1	Trial2	Trial3
Query1	11600	302	26	0.17	6.62	76.92
Query2	12200	329	27	0.16	6.08	74.07
Query3	34100	2626	77	0.06	0.76	25.97
Query4	30040	2043	68	0.07	0.98	29.41
Query5	11250	281	25	0.18	7.12	80.00
Query6	10050	231	23	0.20	8.66	86.96
Query7	20050	902	45	0.10	2.22	44.44
Query8	12300	344	28	0.16	5.81	71.43
Query9	15000	510	34	0.13	3.92	58.82
Query10	12550	351	28	0.16	5.70	71.43
Query11	22520	1149	51	0.09	1.74	39.22
Query12	15230	518	34	0.13	3.86	58.82
Query13	14520	479	33	0.14	4.18	60.61
Query14	14520	479	33	0.14	4.18	60.61
Query15	10050	231	23	0.20	8.66	86.96
Query16	18620	782	42	0.11	2.56	47.62
Query17	18520	778	42	0.11	2.57	47.62
Query18	26380	1556	59	0.08	1.29	33.90
Query19	28600	1830	64	0.07	1.09	31.25
Query20	12500	350	28	0.16	5.71	71.43

Table 6.6 Number of results and precision for three trials on CAIA

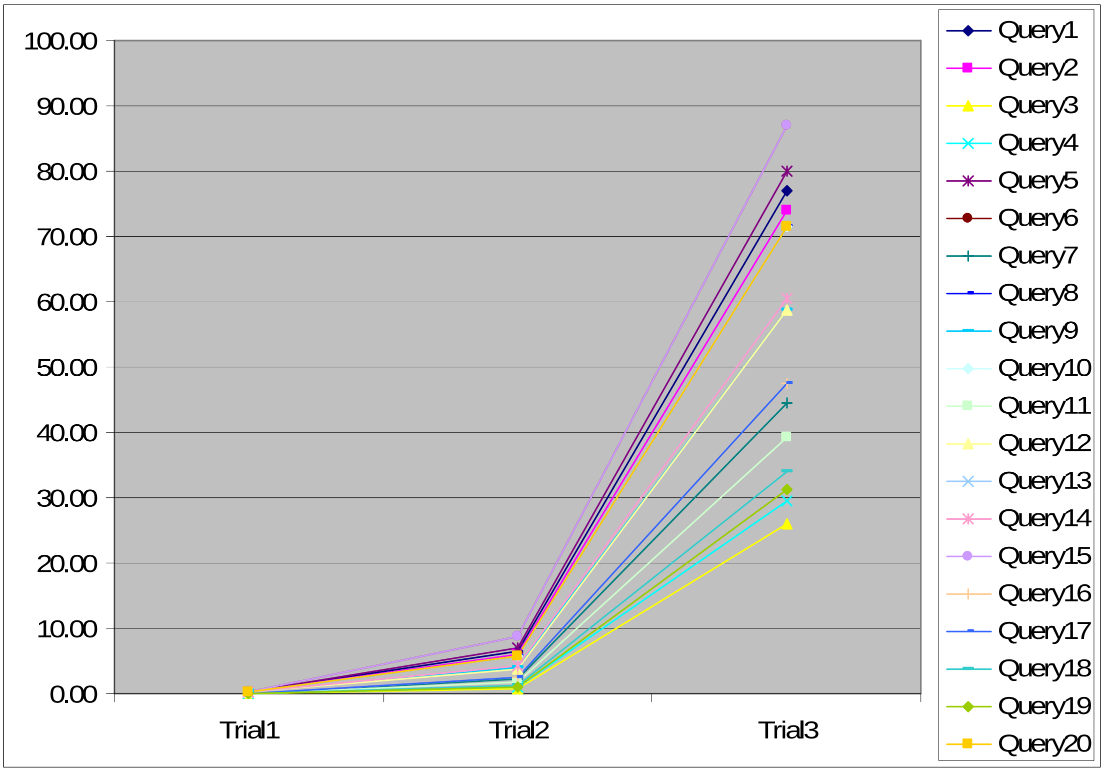


Figure 6.3 Chart shows how the precision of the query increases over trials on CAIA

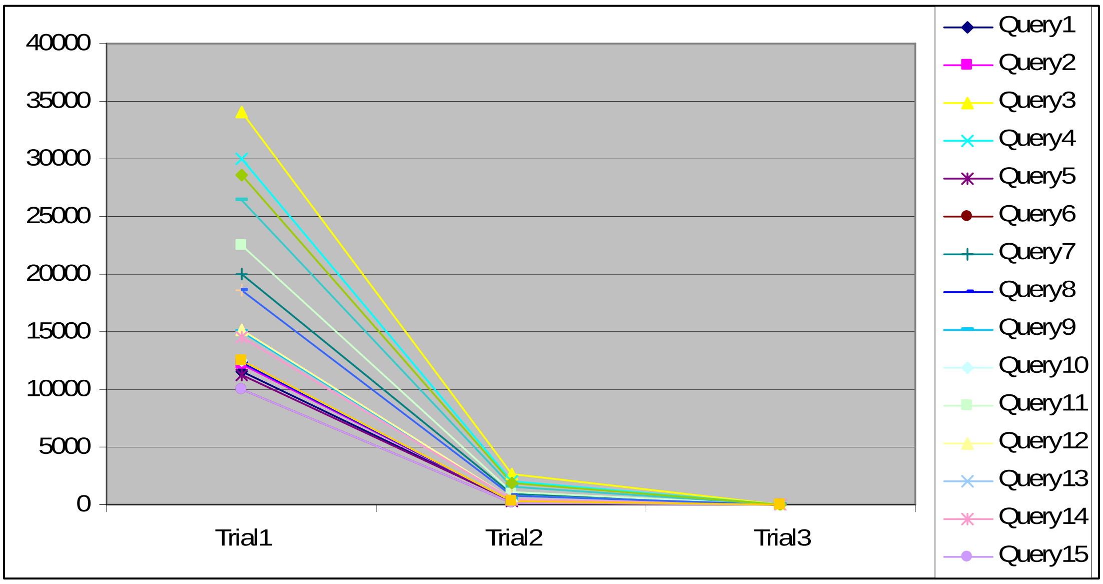


Figure 6.4 Chart shows how the number of results decreases over trials on CAIA

Also, the order of the search result was improved. The following chart shows that clearly.

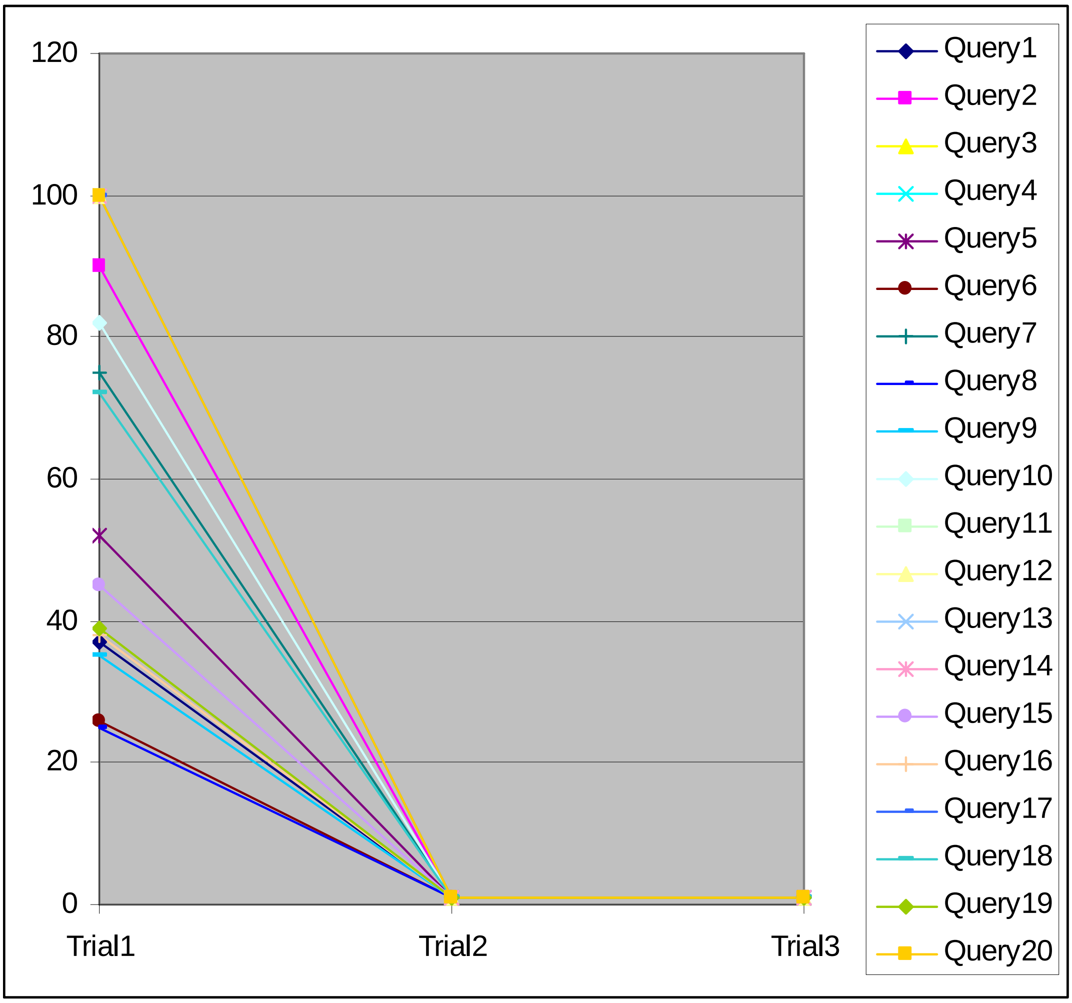


Figure 6.5 Chart shows how the order of the retrieved result is improved over trials

III. THE ANALYSIS OF THE ADAPTABILITY

We have the UP for the user, which stores all URLs he visited, and assign keywords for each URL. We want to measure how CAIA is being able to adapt contents of the UP over time. In addition, we want to measure how good is the correlation between the keywords assigned to the URLs in the UP and the context of the URLs. We used the same formulas as in [8] that define a Fitness value to show the correlation between the weights of the URL's keywords calculated automatically by the CAIA (*T)* and the weights of the URL's keywords calculated by the user (*S*), as follows:

Weight calculated by the user: $S_{URL} = \sum_{k=1}^{m} b_k . W_k$ Equation (3)

Where W_k is the weight of attribute *k*, and b_k equals to 1 if the user judges the keyword *k* in the URL, as relevant for the context of the URL, otherwise b_k equals to zero.

Weight calculated by CAIA: $T_{URL} = \sum_{k=1}^{m} W_k$ Equation (4)

Then, we define the Fitness value, which reflects the correlation between the two adaptations for URL_j:

Fitness value: $F_j = S_j / T_j$ Equation (5)

We have entered fifty different queries. After frequent interactions of retrieval, we checked the correlation of each keyword with the context of the URLs in the UP and calculated *S* and *T* values, and then a Fitness value was calculated for the keywords of each URL in the UP. The following table and figures show the Fitness values as they are converging over time to each other.

We conclude that CAIA is able to predict and adapt the URL's keywords to reflect well the context of the URLs in the UP over time. It is clear that CAIA needs to be used by the user over time and it then gives better correlation between the URL and the relevant keywords.

URL	S_{URL}	T_{URL}	F_j
URL1	1.5	2.1	0.71
URL2	3.5	4.5	0.78
URL3	2.1	2.6	0.81
URL4	3.8	4.8	0.79
URL5	1.9	1.9	1
URL6	1.8	2.8	0.64
URL7	2.8	3.8	0.74
URL8	3	4.5	0.67
URL9	2.1	2.7	0.78
URL10	2	2.6	0.77
URL11	2.8	2.8	1
URL12	5	6	0.83
URL13	2.3	2.3	1
URL14	2.5	2.5	1
URL15	2.5	2.5	1
URL16	2.8	2.8	1
URL17	4.5	4.5	1
URL18	1.5	1.5	1
URL19	2.5	2.5	1
URL20	3.9	3.9	1

Table 6.7 Values of interest calculated by the user and by CAIA and the Fitness value

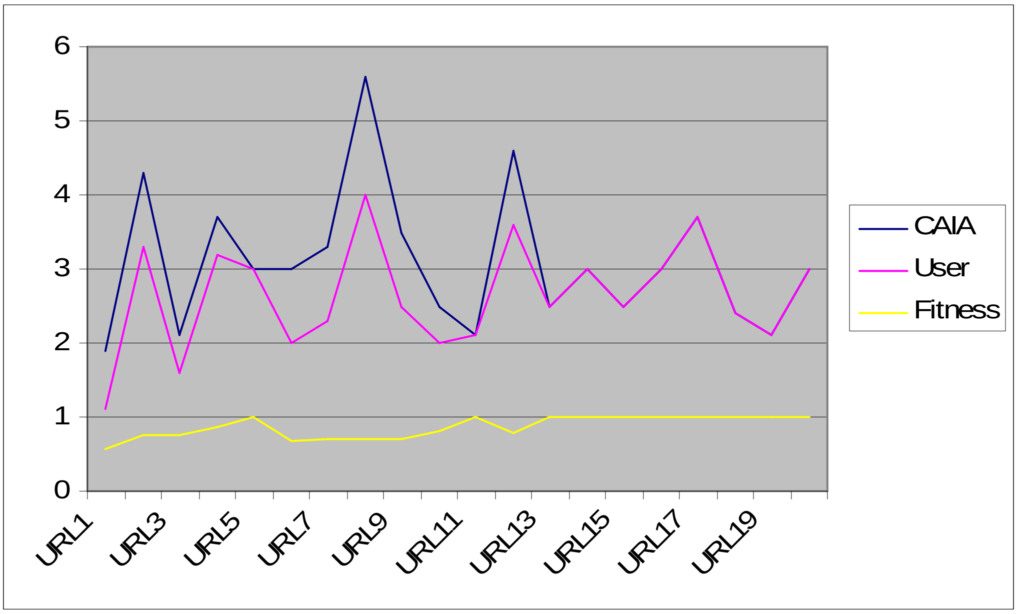


Figure 6.6 2-D chart of the values of interest calculated by the user and by CAIA

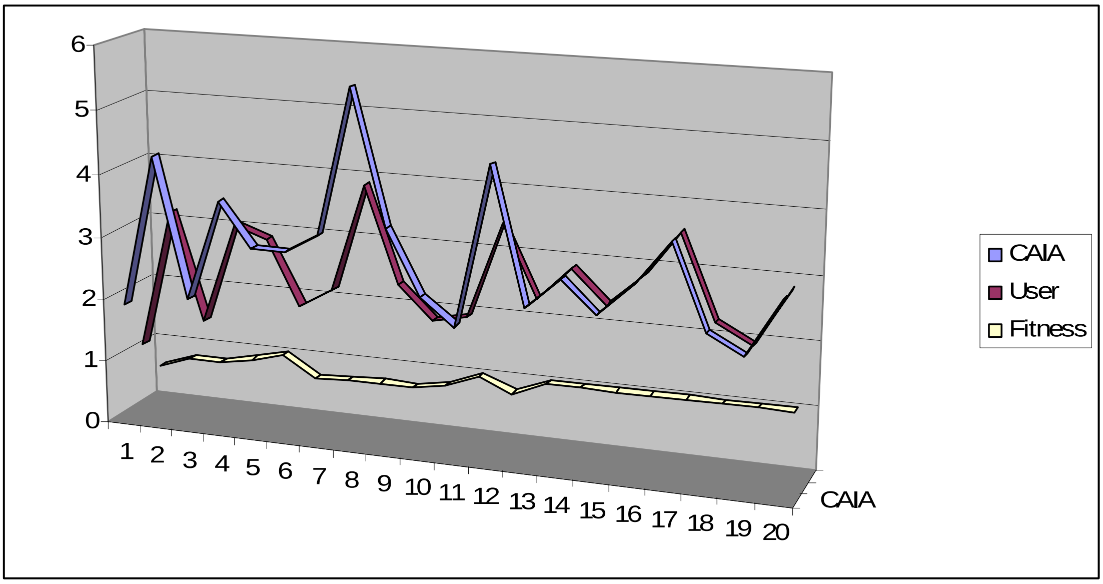


Figure 6.7 3-D chart of the values of interest calculated by the user and by CAIA

CHAPTER 7

CONCLUSION AND FUTURE WORK

I. CONCLUSION

We conclude the thesis with the benefits we found in the developed system, CAIA (Collaborative Autonomous Interface Agent).

PERSONALIZATION

The system is developed in a way that it is adaptable. It can be customized based on the user's preferences. An important example is the weight of the factors that affect the calculation of relevant Websites to the user. A list of factors, such as printing, adding to favorite, saving … etc., that are implicit factors can be ranked based on the user's preferences.

Another important example is the user's profile. The system learns and adapts the user's profile (UP) from the user's search history and browsing behavior respectively. The system uses UP to find relevant information to the user. The system uses both explicit and implicit feedbacks to capture and adapt user's preferences overtime. The profile is tuning to the user's behaviors and upgrading itself to be more powerful and helpful to the user.

COLLABORATION

The system have some autonomy and the ability to sense and react to its environment, as well as socially communicate and cooperate with other agents in order to accomplish its duties, which are delegated from the customer. In a collaborative agent model, a particular agent may not have any prior knowledge, while there may be a number of agents belonging to other customers which do. Instead of each agent re-learning what other agents have already learned through experience, agents can simply ask for help in such cases. This gives each agent access to a potentially vast body of experience that already exists. Over time each agent builds up a trust relationship with each of its peers analogous to the way we consult different experts for help in particular domains and learn to trust or disregard the opinions of particular individuals. The collaboration feature in the system guarantees the privacy of the user so it is sharing only the common records of the user. The user can set any Website to be private while he is browsing it.

The collaboration increases the performance of the agent by improving the communication overhead, instead of communicating to the Internet to get the search results; it can be imported from another agent in the same network. Also, it increases the precision of the search results for the agent get help from other agents.

COMMUNICATION OVERHEAD, TIME AND EFFORTS

One of the main benefits of the system is improving the communication overhead. Instead of connecting to Internet and search for keywords and then get many results that will take time and then browse through them to get the correct one, we get the relevant results directly from the agent. This will increase the performance of the agent and will decrease the time of the search and getting the search results. Also, the user’s efforts will be decreased. So, instead of browsing the huge number of the search results, s/he will get the correct results in a fast way without more efforts to browse that big number of results in the normal case.

PRECISION

Precision is how much results related to the user. The precision of the system is high compared to the most famous search engine, Google. The reason behind the high precision of the system is because the agent is monitoring the user’s interactions with previous searches. Also, it upgrades the user’s query by adding relevant keywords from previous history in the agent’s database. So, we are using query reformulation which means an explicit assistance by the system formulating related and narrower queries; both to aid searching by refining and further detailing underspecified queries, this means the query that will be forwarded to the Meta search engine will be completely different than the one submitted by the user. Examples of the refining processes can be: Spell checker for the user's query terms and noise words removal of the user's query which removes any noise words such as "*the*", "*where*" etc. Noise words are saved in the UP so that s/he can extend them and add more words that s/he feel they are going to make his/her search results worse.

II. CONTRIBUTIONS

The contributions of this thesis can be summarized in the following points:

- The literature of similar efforts was studied and we identified the limitations of the related works.
- A new customized process was proposed and it uses personalization approach in improving the search process.
- A survey was conducted to determine the users' behaviors which reflect their interests. The survey results were analyzed and we concluded a set of weights assigned to those behaviors.
- A collaborative autonomous interface agent was designed and fully implemented using Java. It personalizes the search process in the Internet, improves it automatically by keeping track of history behaviors and helps community's members to share their browsing and searching history. Also, we have evaluated and tested this system.

III. FUTURE WORK

The developed system has many features and provides a great help the user in his/her search in the Internet. However, there are some areas of improvement as following.

AUTOMATIC DETECTION OF USER'S BEHAVIORS' WEIGHTS

We conducted a survey to get the user's behaviors and their priorities as explained in Chapter-3. We set the default weights of those factors that affect the user's interest according to the values we got in the survey. We also provide the user the ability to change his/her preferences in those factors. However, it is good in the future to let the agent itself detects the frequent of the user's behaviors and calculates in some way the weights of the those factors that affect the user's interest in Websites. For example, if the system observes that the user always prints a Website that he visits with certain time, it should change the weight of the print factor and make it higher. The change of the weights of the factors should by dynamic and handled by the system to make it more efficient.

AUTOMATIC DETECTION OF NOISE WORDS

We defined a big list of noise words in the UP in the database. The system checks the user's query against those noise words and removes any word in the user's query if it is found. The user can extend them and add more words that s/he feel they are going to make his/her search results worse. However, it is good in the future to let the agent itself detects the frequency of the new noise words in the user's query. It would be better if the system automatically detects those words in some way and adds them to the noise words list.

CLUSTERING THE AGENTS

We developed the system to be collaborative as the agent could get help in the search from other agent. We assumed that all agents are from the same community and hence we did not distinguish between different users' interests. However, it is good in the future to cluster the agents based on their interests so that the agents with similar interests could collaborate. It will be more efficient to the user if s/he gets help from another agent which has similar interests to his/her interest. The user should define his/her interests when s/he setups the system and s/he can later update his/her interests.

UTILIZING OTHER SEARCH ENGINES APIs

We use Google search engine APIs in order to get more results to the user's query after we refine it. However, it is good to utilize other search engines in the future if they provide APIs like Google. After getting the results from more than one search engine, the results could be filtered and re-ordered based on the UP. Our system is taking care of removing any duplicated URL in the search results. This step if implemented in the future, it will add more value to the system and the results will be more comprehensive.

ADDING MORE BEHAVIORS TO BE OBSERVED

In our system, we observe the most frequent behaviors we found while we conducted the survey. However, it is good to observe more behaviors that are not explored in our system. For example, it is good to trace the words the user highlights in the Website. S/he may copy them to be used in other application. Those words should be added to his/her profile and given some weight in order to help him/her when s/he performs a new search.

GETTING MORE ACCURATE KEYWORDS FROM THE WEBSITE

In the developed system, we trace the title of the Website and the summary we get from the search engine. Then, we extract the keywords of the Website from the title and the summary and we add those keywords to the UP with a certain weight and associate them to the Website's URL to help in new search. However, it is good in the future to get more accurate keywords because some Websites' titles are not meaningful. So, it is good idea to trace the Website's content and based on some algorithms extract the keywords of the Website. There are a lot of algorithms in the literature handling this process. For example, there is algorithm that takes the most frequent words in the Website and considers them as keywords of the Website.

MAKING THE AGENT COMMUNICATES WITH OTHER APPLICATIONS

To make the developed agent more efficient, it is good in the future to let it communicates with other applications in order to improve the UP. The goal is to make the system more personalized to the user and the system will be more powerful in learning automatically his/her preferences. For example, it is good if the system could detects the interest of the user from his/her emails and then reflect this on his/her UP. To make this idea more practical, other agents could be developed to play as middle layer between the system and the other applications. Those other agents send to our system the more frequent behaviors of the user that reflect his/her interest.

APPENDICES

A. USER'S MANUAL FOR CAIA

OVERVIEW

CAIA (Collaborative Autonomous Interface Agent) system is intended to personalize the Internet search process in order to improve it. CAIA is developed to reside in the user’s machine not in the Internet to make all private information on his/her machine. CAIA learns the user’s preferences either explicitly or implicitly from his browsing behavior. It then stores them along with the Websites' information in a User’s Profile (UP). UP is containing all visited Websites' information and their relevant keywords. We mean by "relevant keywords" the keywords the user entered when s/he searches and the Websites' keywords gathered from their titles and summary.

FEATURES

- It Improves performance of the information retrieval of Internet search engines based on specified, measurable attributes.
- It is an autonomous, intelligent agent that depends on Meta search engines.
- It monitors the user's actions to prioritize the search results. It learns based on the user's preferences and information content of the search results.
- It learns the user’s preferences during his/her searching and browsing the search results.
- It builds a user’s profile based on his/her preferences. This user’s profile contains all visited Websites with their keywords and any entered keywords by the user in the search of those Websites.
- The agent filters and refines the query entered by the user.

- It filters the retrieved information based on the current user's preferences.
- It collaborates with other CAIA agents for exchanging the search results.

SYSTEM REQUIREMENTS

CAIA is written in 100% pure Java and it can run on any Java 2 platform. For a complete list of platforms that support Java, please check http://www.javasoft.com. System requirements in hardware are basically the same as that of any Java 2 environment, 128 MB RAM is recommendable. Additionally it is recommended to have a fast and stable Internet connection (T1/T3/DSL) and approximately 100 MB of free space on a hard disc (40 MB for the application and 60 MB for the working data). Regarding software requirements, we assume that the user has *Borland JBuilder* with Java version 1.4 or more, *Internet Explorer* 5.x and more and *Microsoft Access* version 2000 or more.

INSTALLATION

The following steps are needed in order to install CAIA:

1. Copy the folder "*caia*" to any place in your system. We will call the desired path that we copy the file is "*caia-path*".
2. Copy the enclosed file "*runjade.bat*" in the "*caia*" directory to your "*jdk x.x/bin*" folder.

UNINSTALLATION

Simply delete the whole directory, where you have installed CAIA, "*caia-path*".

STARTING CAIA IN WINDOWS

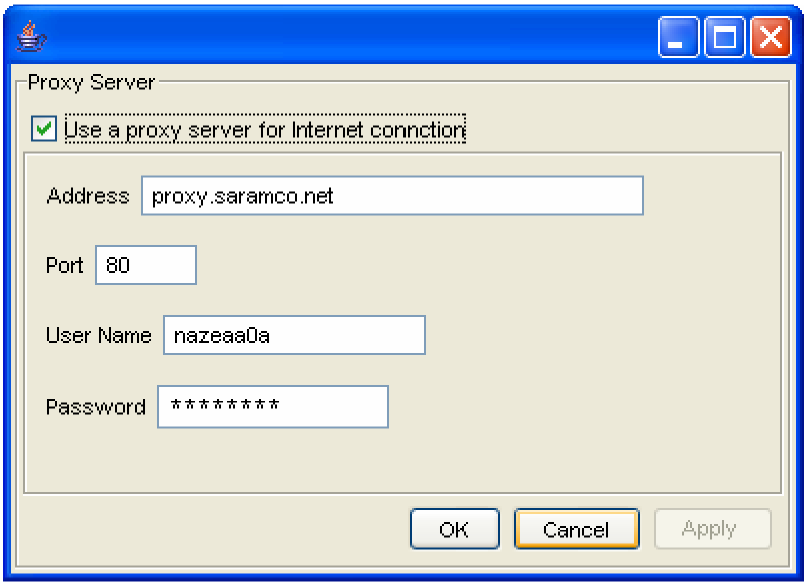
The following steps are needed in order to start CAIA in Windows. Steps 2 and 3 are required only for first time.

1. Double click on the file "*runjade.bat*" in order to run JADE server.
2. Start *Borland JBuilder* and add the JAR libraries. You can do this by clicking on "*Project*" then click on "*Project properties*" then click on "*paths*" on the left then click on "*Required libraries*" tab and you will see "*libCAIA*" highlight it and click on "*Edit*" button on the right. Then, click on "*Add*" button and a popup window will appear browsing the file system. Go to the "*caia*" folder, *caia-path*", browse "*lib*" folder and select all the JAR files to be added.
3. Inside *Borland JBuilder,* go to the running configuration through "*Run*" menu item then select "*Configurations*". You will see some configuration with persons' names like "*JADE_Ahmed*" and you change them by highlighting one of them and click on "*Edit*" on the right, then:
    a. In "Name" text box, enter a name of the configuration.
    b. In "*VM parameters*" text box, put your path for the system as following: -Dpath="*caia-path*"
    c. In "*Application parameters*" text box, put the computer name that runs JADE server and the desired agent name and leave other fields as it as following: "-host *Computer-Name* -container *Agent-Name*:brain.CAIAAgent".
4. Run the configuration with the desired agent name configured in last steps and enjoy the application.

SETTING CAIA FOR FIRST TIME

There are some settings should be visited by the user if it is the first time of using CAIA. Those settings can be updated later.

First, the user should set the firewall proxy settings whether s/he uses it or not. S/he can click on "*settings*" in the menu bar and then click on "*proxy*" then a window will appear to him/her. The user will see a similar popup window as in Internet Explorer. S/he can choose from the first option whether he uses a firewall proxy or not and based on his choice s/he should the below textboxes. If s/he chooses to use a proxy server for Internet connection, s/he will be able to enter the proxy's information including the *address*, *port*, *user-name* and the *password* settings.



Proxy Server

☑ Use a proxy server for Internet connction

Address: proxy.saramco.net

Port: 80

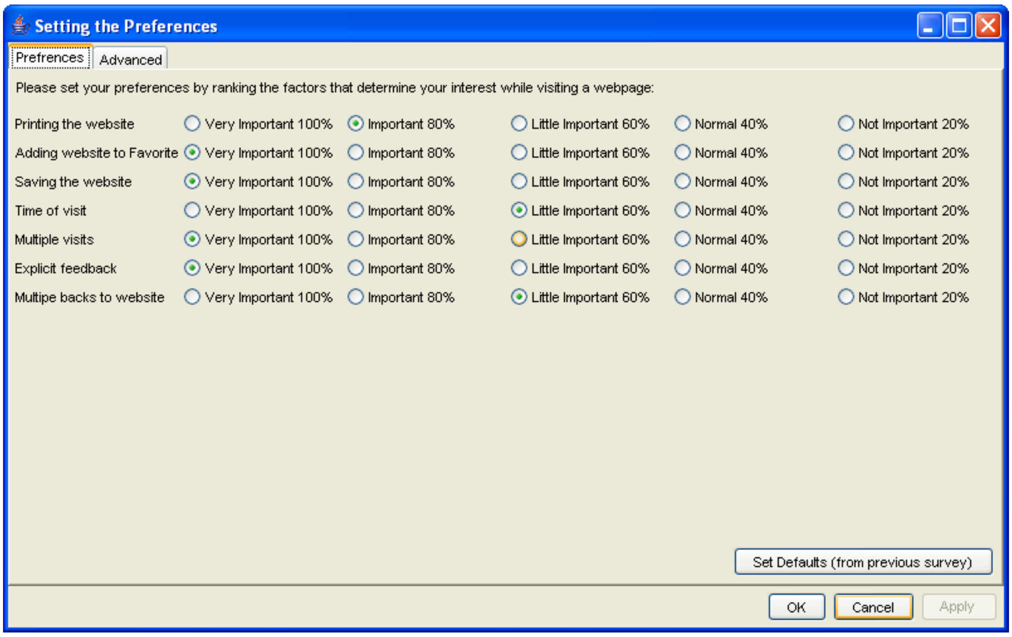
User Name: nazeaa0a

Password: ********

Figure A.1 Setting the proxy

Second, the user should set his/her preferences settings. S/he can click on "*settings*" in the menu bar and then click on "*preferences*" then a window will appear to him/her. The user will see two tabs then s/he can select the first tab "*preferences*". It allows the user to set the weight of different factors that determine his/her interest in a Website.

Here, the user can control his/her behaviors priorities. There is an option to set the default values based on a survey conducted previously and explained in Chapter-3. The following figure shows this tab.



Please set your preferences by ranking the factors that determine your interest while visiting a webpage:

Printing the website	○ Very Important 100%	● Important 80%	○ Little Important 60%	○ Normal 40%	○ Not Important 20%
Adding website to Favorite	● Very Important 100%	○ Important 80%	○ Little Important 60%	○ Normal 40%	○ Not Important 20%
Saving the website	● Very Important 100%	○ Important 80%	○ Little Important 60%	○ Normal 40%	○ Not Important 20%
Time of visit	○ Very Important 100%	○ Important 80%	● Little Important 60%	○ Normal 40%	○ Not Important 20%
Multiple visits	● Very Important 100%	○ Important 80%	○ Little Important 60%	○ Normal 40%	○ Not Important 20%
Explicit feedback	● Very Important 100%	○ Important 80%	○ Little Important 60%	○ Normal 40%	○ Not Important 20%
Multipe backs to website	○ Very Important 100%	○ Important 80%	● Little Important 60%	○ Normal 40%	○ Not Important 20%

Figure A.2 Setting the preferences

Finally, the user should set his/her advanced preferences settings. S/he can click on "*settings*" in the menu bar and then click on "*preferences settings*" then a window will appear to him/her. The user will see two tabs then s/he can select the first tab "*Advanced*". It gives the user the flexibility to set his/her preferences as explained below followed by a figure that shows this tab.

- *Refining Query Options*:
  i. If the user wants to use Google spell-checker for his/her queries.
  ii. If the user wants the system to suggest related keywords to his/her queries based on his/her *Profile*.
- *Monitor Activities Options*:

i. If the user wants the system to monitor all his/her all browsing activities, i.e. when s/he accesses any Website directly without searching for them using the Website's address textbox.

ii. If the user wants the system to monitor all his/her behaviors on the searching results after s/he searches through the system.

- *Searching Options:*

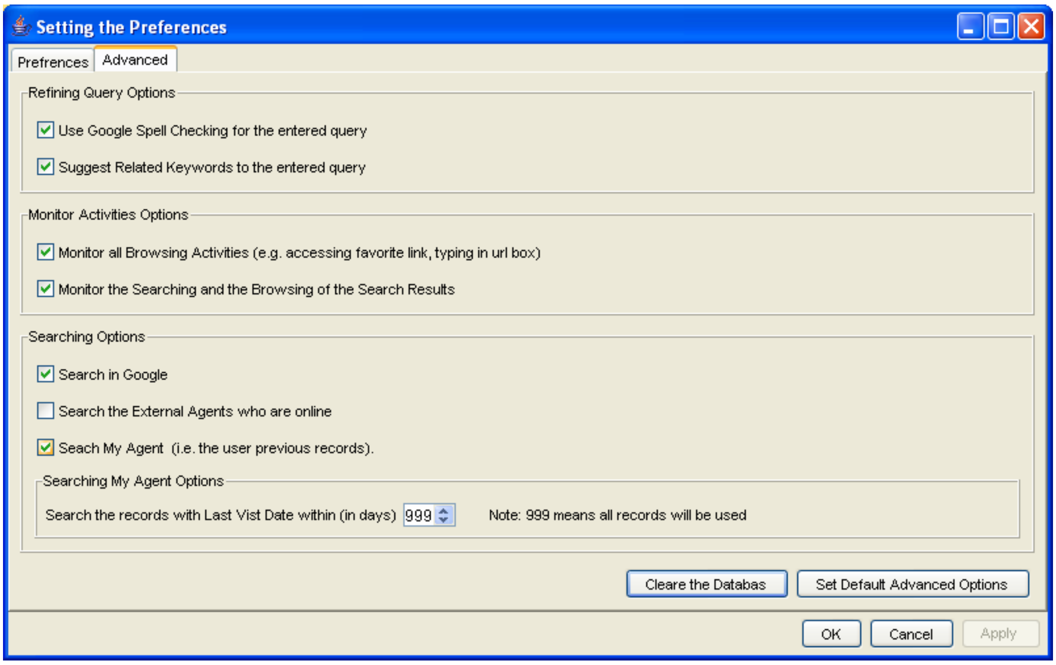
i. If the user wants the system to search in Google using the Google APIs with the refined keywords.

ii. If the user wants to search in other external agents in the community.

iii. If the user wants the system to search in his/her agent based on his/her *Profile*. Also, s/he can set the duration to search within, i.e. s/he can ignore the past history with this option.

iv. If the user wants the system to clear his/her *Profile* (database).

v. If the user wants to set all above options to the default values.



Refining Query Options

☑ Use Google Spell Checking for the entered query

☑ Suggest Related Keywords to the entered query

Monitor Activities Options

☑ Monitor all Browsing Activities (e.g. accessing favorite link, typing in url box)

☑ Monitor the Searching and the Browsing of the Search Results

Searching Options

☑ Search in Google

☐ Search the External Agents who are online

☑ Seach My Agent (i.e. the user previous records).

Searching My Agent Options

Search the records with Last Vist Date within (in days) 999

Note: 999 means all records will be used

Figure A.3 Setting the advanced preferences

WORKING WITH CAIA

The user interface of CAIA is somehow similar to Internet Explorer. It has a menu that contains:

1. "*File*" menu which has some options like "*Save*", "*Print*" and "*Exit*" as shown below in the following figure.

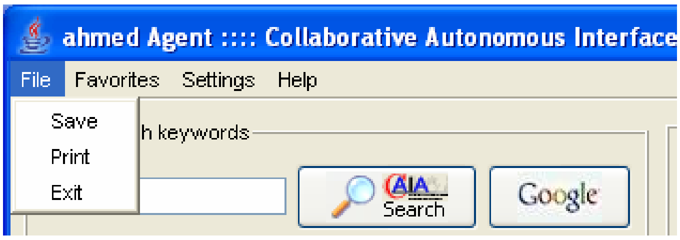


Figure A.4 *File* Menu

2. "*Favorites*" Menu which has two action options "*Add to Favorites*" and "*Organize Favorites*". And then, the user's favorites are listed and if s/he clicks any favorite Website, the browser will open that Website. The menu is shown in the below figure.

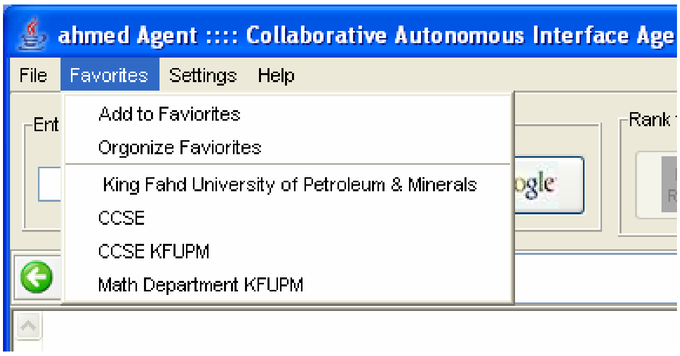
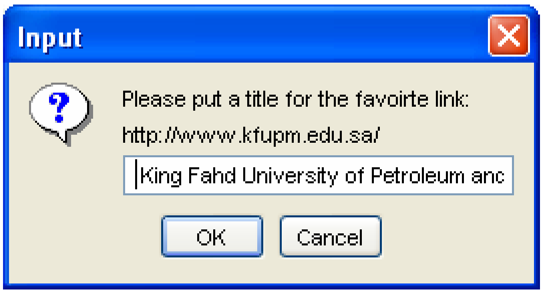


Figure A.5 *Favorites* Menu

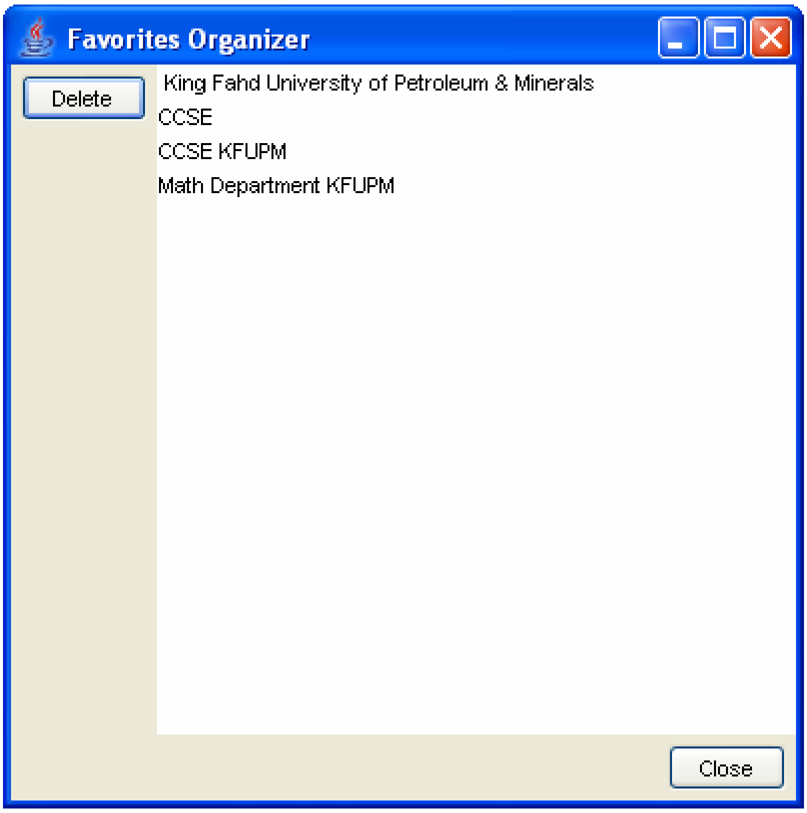
When the user clicks on "*Add to Favorites*", a popup window, dialog, will appear asking him/her to enter the name of the Website to be added in the *Favorites* Menu as shown below in the following figure.



Please put a title for the favoirte link:
http://www.kfupm.edu.sa/
King Fahd University of Petroleum anc

Figure A.6 Favorite addition dialog

On the other hand, when the user clicks on "*Organize Favorites*", a popup window will appear showing user's favorites Websites. The user can choose any one of them and click on "delete" button if s/he does not like it.



King Fahd University of Petroleum & Minerals
CCSE
CCSE KFUPM
Math Department KFUPM

Figure A.7 *Favorites-Organizer*

3. "*Settings*" Menu which has two options "*Proxy*" and "*Preferences*" as shown in the below figure.

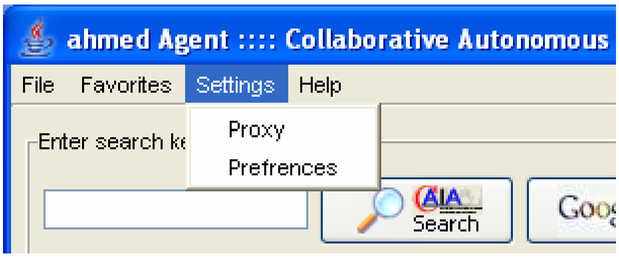


Figure A.8 *Settings* Menu

This menu items are explained and elaborated in the previous subsection "*SETTING CAIA FOR FIRST TIME*".

4. "*Help*" Menu which has only one option for time being "*about*" which gives an idea about the application and the copyright.

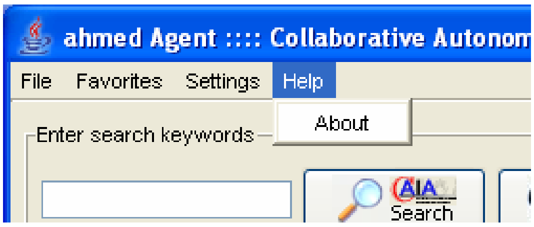


Figure A.9 *Help* Menu

After explaining the menu items, the following figure shows the other parts of CAIA.

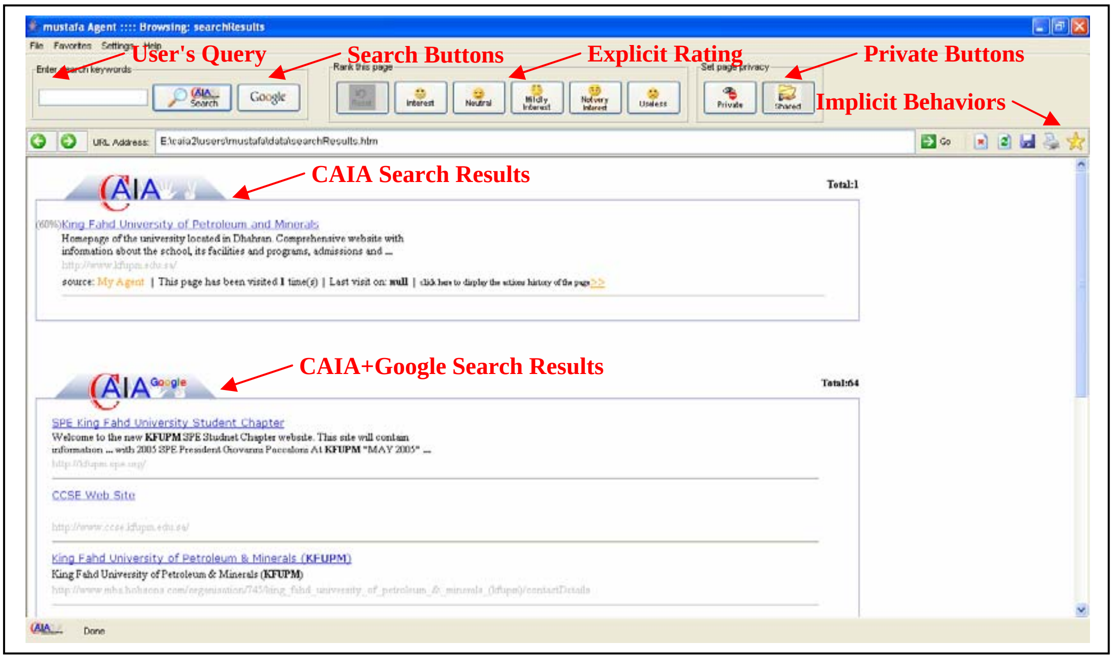
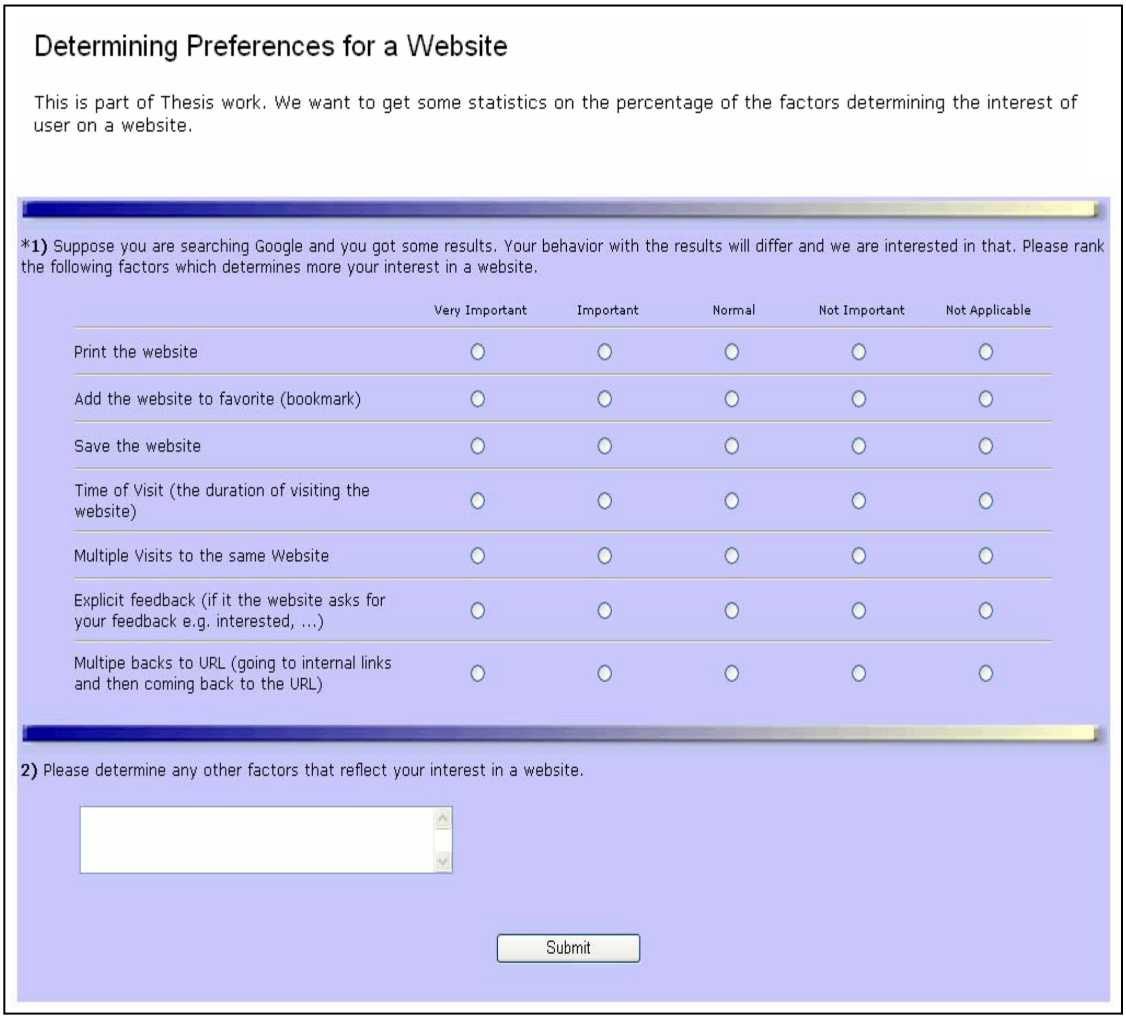


Figure A.10 Other parts of CAIA

B. THE SURVEY FORM



Determining Preferences for a Website

This is part of Thesis work. We want to get some statistics on the percentage of the factors determining the interest of user on a website.

***1)** Suppose you are searching Google and you got some results. Your behavior with the results will differ and we are interested in that. Please rank the following factors which determines more your interest in a website.

	Very Important	Important	Normal	Not Important	Not Applicable
Print the website	○	○	○	○	○
Add the website to favorite (bookmark)	○	○	○	○	○
Save the website	○	○	○	○	○
Time of Visit (the duration of visiting the website)	○	○	○	○	○
Multiple Visits to the same Website	○	○	○	○	○
Explicit feedback (if it the website asks for your feedback e.g. interested, ...)	○	○	○	○	○
Multipe backs to URL (going to internal links and then coming back to the URL)	○	○	○	○	○

2) Please determine any other factors that reflect your interest in a website.

Figure B.1 The survey form

REFERENCES

[1] Chen, L., Sycara, K.: WebMate: A Personal Agent for Browsing and Search. In: Proceedings of the2ndInternational Conference on Autonomous Agents, ACM Press, New York, NY (1998) 132-139

[2] Degemmis, M., Lops, P., Semeraro, G., Costabile, M.F., Lichelli, O., Guida, S.P.: A Hybrid Collaborative Recommender System Based on User Profiles. In: Proceedings of the Sixth International Conference on Enterprise Information Systems, Vol. 4. Porto (2004) 162-169

[3] Dieberger, A.: Supporting Social Navigation on the World Wide Web. In: Int. J. Human-Computer Studies 46 (1997) 805-825

[4] Gori, M., Witten, I.: The bubble of Web visibility. In: Communications of the ACM (2004 In Press)

[5] Somlo, G., Howe, A: Using Web Helper Agent Profiles in Query Generation. In: Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems, ACM Press, Melbourne (2003) 812- 818

[6] Turner, R., Turner, E., Wagner, T., Wheeler, T., Ogle, N.: Using Explicit, A Priori Contextual Knowledge in an Intelligent Web Search Agent. In: Lecture Notes in Artificial Intelligence 2116 (2001) 343-352

[7] Tarek Helmy, Satoshi Amamiya, Tsunenori Mine, Makoto Amamiya,

"An Agent-oriented Personalized Web Searching System", Lecture Notes in Computer Science LCNS (Springer-Verlag), May 2002 pp. 113-116.

[8] Tarek Helmy, Satoshi Amamiya, Tsunenori Mine, Makoto Amamiya, "A New Approach of the Collaborative User Interface Agents", Proceedings of IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT'03), pp. 147-153, October 13-17, 2003, (Halifax, Canada).

[9] Tarek Helmy, Satoshi Amamiya, Makoto Amamiya, "An Agent-Oriented Model for the Next Generation Personalized Web Searching Systems", International Journal of Web Intelligence and Agent Systems, Volume 2, Number 3, 2003, pp. 217-225.

[10] Tarek H. El-Basuny, "A Ubiquitous Approach for Next Generation Information Retrieval System", Proceedings of the IEEE/CS International Conference on Information and Computer Science (ICICS 2004), KFUPM, Dhahran, Saudi Arabia, 28-30 November 2004, ISSN 9960-07-212-6©KFUPM, pp. 501-513.

[11] Tarek H. El-Basuny, "Multi-agent based Electronic Commerce System", Proceedings of the 1st IEEE International Computer Engineering Conference New Technologies for the Information Society, Cairo University, Egypt, 27-30 December 2004, pp. 791-796.

[12] Lieberman H., "Letizia: An Agent That Assists Web Browsing". In Proceedings of the International Joint Conference on Artificial

Intelligence, Montreal, CA, 1995

[13] Mladenic D. , "Text-learning and related intelligent agents". Revised version In IEEE Expert special issue on Applications of Intelligent Information Retrieval, 1999

[14] Gauch S., Chaffee J. and Pretschner A., "Ontology Based Personalized Search." Web Intelligence and Agent Systems (in press), 2004.

[15] Thomas C. and Fischer G., "Using Agents to Personalize the Web." In Proceedings of the 2nd International Conference on Intelligent User Interfaces, Orlando, Florida, USA, 1997, pp.53-60

[16] Pazzani M., Muramatsu and Billsus D., "Syskill&Webert: Identifying interesting Web sites." In Proceedings of 13th National Conference of Artificial Intelligence, Portland, OR, 1996

[17] Tanudjaja F. and Mui L., "Persona: A contextualized and personalized Web search." In Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS'02), Big Island, Hawaii, 2002, volume3 pp. 53

[18] Dumais S., Cutrell E., Cadiz JJ., Jancke G., Sarin R. and Robbings D., "Stuff I've Seen: A System for Personal Information Retrieval and Re-use." In proceedings of SIGIR 2003, Toronto, Canada

[19] Nikolaos Nanas, "Literature Review: Information Filtering for Knowledge Management", KMI Technical Report: kmi-01-16, 2001 <http://kmi.open.ac.uk/publications/pdf/kmi-01-16.pdf>

[20] Kazunari Sugiyama, Kenji Hatano, Masatoshi Yoshikawa and Shunsuke

Uemura: ``Adaptive Web Search Based on User's Implicit Preference," The 15th Data Engineering Workshop (DEWS2004), March 4-6, 2004.

[21] T. Joachims, D. Freitag, and T. Mitchell, “WebWatcher: A tour guide for the World Wide Web.”, In Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence, 1997.

[22] Mobasher B., Dai H., Luo T., Nakagawa M., Sun Y. and Wiltshire J., "Discovery of Aggregate Usage Profiles for Web Personalization", In Proceedings of Web Mining for E-commerce Workshop (KDD-2000), Boston, MA 2000

[23] Alexandros Moukas, “Amalthaea: Information Discovery And Filtering Using A Multiagent Evolving Ecosystem”, In Proceedings of the Conference on the Practical Application of Intelligent Agents and MultiAgent Technology, 1996.

[24] Moukas, A. and Maes, P. (1998). “Amalthaea: An Evolving Multi-Agent Information Filtering and Discovery System for the WWW.” Autonomous Agents and Multi-Agent Systems, 1(1): 59-88.

[25] A. Moukas, G. Zacharias and P. Maes, "Amalthaea and Histos: Multiagent Systems for WWW sites and recommendations". In M. Klusch (ed.), Intelligent Information Agents, Springer, 1999, pages: 293-322.

[26] S. E. Middleton, “Interface Agents: A Review of the Field”, Technical Report Number ECSTR-IAM01-001, University of Southampton, August 2001

[27] Choo, C.W., Detlor, B. and Turnbull, D., “Information Seeking on the Web: An Integrated Model of Browsing and Searching.”, Proceedings of the Annual Conference of the American Society for Information Science, pp.127-135, Washington, DC, 1999

[28] Jansen, J., "Using an Intelligent Agent to Enhance Search Engine Performance.", First Monday 2, no. 3 (March 3, 1997). <http://www.firstmonday.dk/issues/issue2_3/jansen/index.html>

[29] Trajkova, Joana, and Susan Gauch, "Improving Ontology-Based User Profiles", Proc. of RIAO 2004, University of Avignon (Vaucluse), France, April 26-28, 2004, pp. 380-389.

[30] Tarek Helmy, Tsunenori Mine and Makoto Amamiya "Adaptive Exploiting User Profile and Interpretation Policy for Searching and Browsing the Web on KODAMA System", 2000

[31] JADE (Java Agent DEvelopment Framework): http://jade.tilab.com/

[32] Google Web APIs (beta): http://www.google.com/apis/index.html

[33] The Wintertree Thesaurus Engine class library: http://www.wintertree-software.com/

[34] EZ JCom: http://www.ezjcom.com/

[35] FIPA: The Foundation for Intelligent Physical Agents http://jade.tilab.com/papers/JADETutorialIEEE/JADETutorial_FIPA.pdf

[36] http://www.fipa.org/

VITAE

My name is Ahmed Ali AlNazer from Saudi Arabia. My address is P.O.Box.5945 Dammam 31432 Saudi Arabia. My emails address is alnazera@hotmail.com.

I have graduated from King Fahd University of Petroleum and Minerals (KFUPM) in 2001 and I hold B.S. degree in computer science with a second honor. My senior project "e-trip portal" was awarded as one of the best university projects in the 2001 annual exhibition.

Since 2001, I am working in the IT of the largest oil producer company in the world, Saudi Aramco. It gave me the exposure to real world information technology deployments. While working in the company, I was pursuing my M.S. degree in computer science as a part time.

I started my master degree study in 2003 and I could complete all the requirements of the master degree by January 2006. My research areas are in artificial intelligent and in software engineering.