INTELLIGENT WEB AGENT

A BTech Project Report Submitted
in Partial Fulfillment of the Requirements
for the Degree of

Bachelor of Technology

by
Sandeep Kumar

to the
Department of Computer Science and Engineering
Indian Institute of Technology Guwahati.
April, 2004

Abstract

Searching has become the most important activity of a typical user in the internet. The internet as of now is highly disorganized and it is nearly impossible for a user to get the webpage he wants. Existing general purpose web search engines help him but throw hundreds of matches to his query which results in only partial solving of his problem. He still has to search for the webpage of his interest among the search results.

Thus a new machine learning technique is developed in this report to learn about the interests of a user based upon his browsing habits and generate a profile for the user. This is done unobtrusively without the user actively providing any relevance feedback as most users are not interested to do so. This profile forms the base to modify and rank the results of a general purpose search engine so that the newly ranked results are more relevant to the user's interests.

The personalized intelligent web agent so developed helps the user in information retrieval from the web with intelligent query formulation using the trigger pair model, clustering of search results to interests, learning interests and finally ranking the results based on interests and displaying them.

Acknowledgements

I would like to express my deepest gratitude to my research supervisor Dr. S. B. Nair in this undergraduate thesis work for providing me the inspiration to work in the field of software agents on the web and providing his valuable support and guidance during the course of the project.

I wish to thank my classmate Karan Singal for being my partner in a previous project on "Automatic Taxonomy Generation" which laid some of the foundations of the present project. I also thank him for testing the present project.

I thank my senior Nandan Chaturbhuj whose undergraduate project on "Intelligent Web and Desktop" inspires the present work and am indebted to his work for a lot of common features incorporated in the present project.

I enjoyed discussions with my colleagues of the "AI" group: Amit W. Chawre, Abhinav Sarje, Ankur Chauhan and Shailendra S. Rathore. They also helped me in testing the present project.

Contents

Chapter 1

Introduction

Syntactically **"intelligent web agent"** [1, 2] can be divided into 3 words namely intelligent, web and agent thus the literal meaning will be a agent whose domain of work is the world wide web and is intelligent. So what's an agent? "An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors."[3].For a web agent the environment is the World Wide Web or the internet as it is commonly known. It perceives it using words of an HTML document acquired from software sensors that connect to the internet using HTTP. The agent's actions will depend on the goal of the agent like for a search agent it would be seeking a website containing related information about the search string. Now we need to add intelligence to this web agent. Defining intelligence itself is large debate in the field of Artificial Intelligence. Basically we can say that a web agent is intelligent if it makes a rational decision when given a choice. In other words, given a goal, it will make decisions to follow the course of actions that would lead it to that goal in a timely manner. After all, the "intelligent" thing to do when confronted with a problem is to work towards a solution and not away from it.

We have added one more property which the intelligent web agent should exhibit which is **"personalization"**. So we need to understand what a Personalized Intelligent Web Agent is. A Personalized Intelligent Web Agent is a software tool which in due course of time learns the user's preferences and behavior and displays ample intelligence in helping him out in his interactions with the World Wide Web. We stress on the word personalization as learning about the user preferences plays the most crucial part in determination of the success of the agent and thereby its greater usage.

1.1 The Problem

The problem we are trying to attack using Personalized Intelligent Web Agent is that of providing user with relevant links to documents available on the internet matching his interests on a particular topic. The problem has gained importance in recent time due the explosive growth of internet which has led creation of an enormous repository of knowledge. This great collection of knowledge has to be used efficiently for faster progress and growth of mankind but due to the heterogeneous nature of information and underlying complexity in the organization of the information, it has become very difficult to find information to one's desires leading to inefficient usage of this huge knowledge base. So what's the use of such a huge knowledge base when one can not find solution to his question (although it exists there!)? Before delving deeper into the problem of searching solution to one's questions in the internet we need to better understand the structure of the internet.

Internet as it is today is a huge collection of hyperlinks on pointers to HTML documents. The hyperlinks themselves are embedded in the HTML documents clicking whom the browser (a software tool to view HTML documents and browse through the internet through hyperlinks) displays the document to which the hyperlink points to. Every document present on the internet has an address in the form of a unique name or URL (Uniform Resource Locator). The problem is finding the document containing information one seeks is in finding the unique URL of it which of course is not easy to guess. Search engines like Google, Yahoo, MSN, etc helps the user in finding documents by fielding a query consisting of a few words representing the information which he wants to retrieve from the internet. The search engine matches the query to its own database of information about the web and returns documents containing words given in the query. Now as the internet has grown too huge so the result returned normally consists of thousands of matching documents and it's impossible for the user to navigate through each of them. The inefficiency of the search engines get magnified as a normal user normally provides two or three words in his query which provides too less information to the search engines to know what the user really wants to look at and a part of the problem resides in the language used too which is English. In English the meaning of words depends on context and usage which is difficult to be determined by software. Therefore in most cases nowadays, these search agents are unable to help the user by providing them documents matching their interests. As the problem is growing with the tremendous growth of information made available on the internet, the need of a Personalized Intelligent Web Agent is also growing which can provide documents which match the user's interest and preferences more than the web based search engines.

1.2 How is it possible?

It's possible because of the simple reason that Personalized Intelligent Web Agents have more access to information about the user than the web based search engines hence they can generate results which match more closely to the user's interests than any search engine. The agent acts as a monitor to all actions taken by the user over the internet. It monitors all the pages he open, all the hyperlinks he clicks and similar actions which provide information about his interests. All these points are taken into consideration and a profile [4] for the user is created containing words which represent his interests to certain accuracy. Given the fact that most of us have more than one interests and in most cases more than one profile, multiple interests for multiple profiles are created depending upon user's demand. Thus having a closer access to user's browser habits, a Personalized Intelligent Web Agent is able to present web documents to a user which match his interests more closely then any available web based search engine.

Chapter 2

Background

Recently, three general approaches have been taken to increase Web search accuracy and performance. One is the development of *meta-search engines* [5] that forward user queries to multiple search engines at the same time in order to increase the coverage and hope to include what the user wants in a short list of top-ranked results. Examples of such meta-search engines include *MetaCrowler* [MC], *Inference Find* [IF] and *Dogpile* [DP]. Another approach is the development of *topic-specific* search engines that are specialized in particular topics. These topics range from vacation guides [VG] to kids health [KH]. The third approach is to use some group or personal profiles to personalize the Web search. Examples of such efforts include *GroupLens* [Konstan et al. 1997], *PHOAKS* [Terveen et al. 1997] among others. The first generation meta-search engines address the problem of decreasing coverage by simultaneously querying multiple general-purpose engines. These meta-search engines suffer to certain extent the inherited problem of *information overflow* that it is difficult for users to pin down specific information that they are searching for. Specialized search engines typically contain much more accurate and narrowly focused information. However it is not easy for a novice user to know where and which specialized engine to use. Most personalized Web search projects reported so far involve collecting user's behavior at a centralized server or a proxy server. While it is effective for the purpose of E-Commerce where vendors can collectively learn consumer behaviors, this approach does present the privacy problem

The clustering, user profiling and other advanced techniques used by these search engines and other projects such as [Bollacker et al. 1998, Bollacker et al. 1999, Lawrence et al. 1999] are *static* in the sense that they are built before the search begins. They cannot be changed dynamically during the real-time search process. Thus they do not reflect the changing interests of the user at different time, in different location or for different subjects. The *static nature* of the existing search engines makes it very difficult, if not impossible, to support the *dynamic changes* of the user's search interests. The augmented features of personalization (or customization) certainly help a search engine to increase its search performance; however their ability is very limited. An intelligent search engines should be built on top of existing search engine design and implementation techniques. It should use the search results of the general-purpose search engines as its starting search space, from which it would adaptively learn in real-time from the user's browsing habits and relevance feedback to boost and to enhance the search performance and the relevance accuracy. With the ability to perform real-time adaptive learning from monitoring of user’s browsing habits and relevance feedback, the search engine is able to learn user's search interest changes or shifts, and thus provides the user with improved search results.

Although dynamic search engines and similar projects like Webwatcher [6], WebMate [7], Letizia [8] and others exist, each uses a different ranking algorithm and normally

uses the current page as the source of user interests and does not take his past browsing habits into consideration. Webwatcher is a tour guide for the web learning from user feedback and hyperlinks of the pages the user has visited. WebMate tries to find the relevance of similar documents available on the internet with the current document the user is viewing by calculating Tf-Idf and finding the similarity among the vectors. The more similar the vectors, the more highly the page will be ranked. Letizia can recommend nearby pages by doing a look-ahead search on hyperlinks. Syskill and Webert [9] is a software agent that learns to rate pages on the web by a three point relevance feedback from the user. Applications like WBI [10] from IBM which supports features like personal history, shortcuts, page watching and traffic lights, MetaBot which searches the web by performing simultaneous query on multiple web search services. Firefly uses software agents that automate the process of retrieving data from the Web based on what they know about their user's tastes and interests. Their core technology is the social filtering (or collaborative filtering). WiseWire uses advanced neural net technology and adaptive collaborative filtering to filter all types of digital content that is personally relevant to the user.

Chapter 3

The Algorithm

After going through the above mentioned researches going on in similar fields, I felt that all the above mentioned approaches are not able to store the information about the user's interest in a correct and efficient way which can be used to rank the results of a web based search engine according to the user's interests. Thereby a new way has represent the knowledge about the user's interests has been proposed here which can be easily used to rank the results using already developed ranking algorithms.

3.1 Definition of Interest

Interest form the basic knowledge block in the profile generated by our algorithm. The user's interest is represented by a group of ten keywords, each having its own weight representing the importance of the keyword in defining the user's interest in the particular topic.

For example lets take a user who is interested in the current Iraq tension will have a interest : said 16, yesterday 15, city 15, sadr 14, iraq 12, holy 12, news 12, east 11, talks11 , tension 11. This gives a lot of information about the user's interest in what "*sadr said yesterday* in a *holy city* of *iraq* and his *talks* about *tension* and *east"*. Here the weights of the words are almost equal hence we are unable to clearly decide the relative importance of words which is better reflected in the following interest: studs 19, Sandeep 15, kumar 11, 2004 6, iitg 6, yahoo 5, iit 5, home 5, browser 4, design 4. In this interest its clear that studs, Sandeep and kumar form the important words while iitg, yahoo, home etc are words with relatively lesser weights in forming the interest.

Now after the representation of interests is clear, two problems arise:

1. Generation of the group of words representing the interest.
2. Creating a profile or knowledge base based upon the interest.

3.2 Generation of Interest

So first discuss how will be generate the words representing one's interest. An important point to consider here is that the process should be as unobtrusive as possible i.e. the process should require as little or no user feedback to generate the set of keywords representing his interests. In the approach used here, the web agent acts as a proxy for the user's browser. Thus all data that is exchanged between the user and the internet is monitored by the agent without the user knowing about it. The agent monitors the user's browsing habits by using the text of the web pages the user has browsed through in

determining his interest. Thus the ten words representing his interest are extracted from the very pages the user browses through which is very logical. So the now the problem reduces to finding the most important words in the pages the user has visited. Presently we discuss how to find the most important words in a HTML document but before it lets have a look into the constitution of a HTML page.

To understand the idea behind generating keywords from a webpage, first we need to understand the structure of a HTML page. A document written in the Hyper Text Markup Language or HTML consists of text which is embedded inside HTML tags which gives special meaning to the text. Some of the important HTML tags are title, meta-names, bold, etc. The text inside *title* tag specifies the title of the HTML document. The author of the HTML document tries to provide the summary of the document in its title hence this tag is most important in considering keywords representing the page. Next in importance come meta-names which consist of a group of words to be used by search engines to identify the keywords of the document. Next come text with large fonts like Heading6, Heading5, etc. The text that is italicized, bold-faced, block-quoted, etc are emphasized text hence has been given relative weights. Then every word that has appeared in the document is given a weight of one to find out words which are repeated many times in the document.

When the user request's a web page to our Personalized Intelligent Web Agent (PIWA), the PIWA behaves as a proxy for his request and downloads the requested webpage and passes a copy of it the browser which is hence viewed by the user. A copy of the same page is parsed by the HTML parser of the PIWA and text forming the title, meta-names, text which is boldfaced or strong, italicized or block-quoted or having a larger font are identified.

The words appearing inside the text are given various weights based upon the relative importance of the HTML tag. For example words of the title are given a weight of 10, words of meta-names are given a weight of 6, those block-quoted are given weight of 4, those boldfaced or underlined are given a weight of 2, those who have the font size of Heading 6 are given weight of 6 and so on.

Words that don't lie within any of the HTML tags are given a weight of 1. Words with size less than three are ignored as these are commonly used words or verbs. Stop word elimination is done to remove some other commonly used words. The weights are summed up for all the words that are present in the webpage. These weights are then sorted and then the top 10 keywords i.e. top 10 words having the maximum value are selected and they form the keywords for the page.

This way we are able to generate information about a user's interest in a very unobtrusive manner. In no case is the user asked to provide any kind of relevance feedback. The whole process is automated and works in the background while the user peacefully browses though the internet unaware of the whole process. This is the basis of whole proposed algorithm which can be defined as '**representation of one's interest in the**

form of a group of words which makes web searches easy and more accurate for a search engine.'

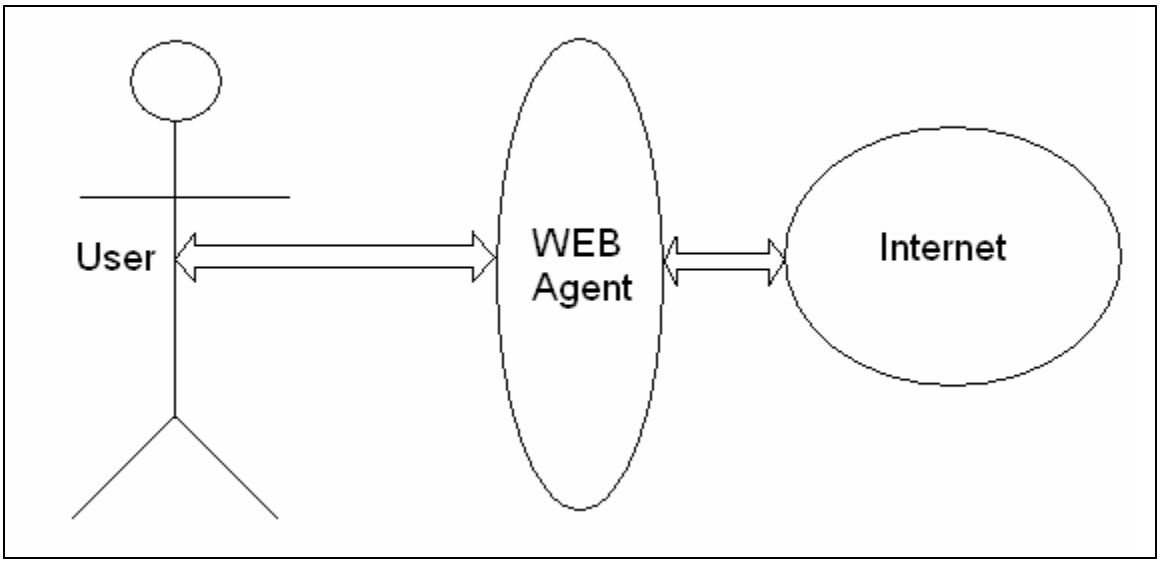

Fig 1: The web agent acting as a proxy between user and internet

3.3 Creation of Profile

Thus whenever the user browses through a webpage, a list of ten keywords is generated for the page. Now we need to use these keywords to create a profile for the user. Here we have two cases:

1. The topic of the page currently viewed is similar to pages visited in the past.
2. The topic of the page currently viewed is new and never visited in the past.

In the former case, the keywords generated in the current page should in a way update the user's already formed interest in the topic. In the latter case the keywords should be appended in his list of interest hence a new interest in created. The latter one is easy to do but how to do the former one.

Let's start from the scratch i.e. the user has no interests listed in his list of interests. The first page he views returns a list of keywords which get appended to his empty list of interest hence creating a new interest. Then he views another page. This page too upon parsing returns a list of keywords. Now these keywords are matched with the already formed interest. If three words or more matches then the new list of keywords is used to update the interest else the keywords are appended at the end creating a new interest.

To update an interest, we create a combined list of words from the matched interest and keywords of current page. In this combined list, the matching words have their weights summed up. Thereafter, the top ten words having the maximum weights form the new

updated interest. Thus this process updates the already formed interest with the new set of keywords.

3.4 Maintenance of profile

As the user goes on surfing, his list of interests will go on increasing. We need to maintain the list so that only important interests are kept. This result's in a small list of interests which is helpful in reducing the time required of matching and updating the list of interests and makes the time requirements nearly invisible to the user.

At present the size of list is fixed at 20 interests. The problem arises in determination of which interest in important and which is not. Hence we keep a timestamp along with each interest. This helps us to determine whether it's a new or an old interest. When our list of interests crosses the mark of 20 interests, we use the combination of timestamp and sum of the weights of the keywords of the interest to mark its importance. Interests having older timestamp or low sum of weights are more likely to the removed from the list on interests.

3.5 Use of profile for web searches

Now after we have generated a list of keywords with attached weights representing various interests of the user, we need to use this acquired knowledge to help the user in his web searches as that's the final aim of the algorithm.

On receiving a search query from the user, PIWA first tries to match the words in the query with already created interests in the list of interests [11]. Three possible cases may arise:

- The query matches one interest.
- The query matches more than one interest.
- The query does not match any of the interest.

If the query does not match any of the interests, PIWA is unable to help the user and simply requests google search to provide with the top 10 matches for the query.

In case the query matches more than one interest, then we sum up the weights for the words in the interests which match the words in the query. The interest which results in the maximum sum of weights is chosen to be the most matched interest. For the case, the query matches one interest, then the interest in the only matched interest.

Once we have a matched interest, we first try to expand the query with the trigger pair model. In this case we take a word having a weight lesser than that of the word in the query having least weight in the interest. This is done to avoid overshadowing of the query with a more popular/weighted word. To explain it better lets take the past example interest: studs 19, Sandeep 15, kumar 11, 2004 6, iitg 6, yahoo 5, iit 5, home 5, browser

4, design 4. Suppose the search query is *Sandeep* and its matches the above interest, its better to append *kumar* (a word having lesser weight than Sandeep) than *studs* to the query as *studs* being a more popular word will pull the results of google more towards itself hence overshadowing the original query *Sandeep*. For query length of one word, one word is appended, and for query of length two or more words, two words are appended for the same reason as explained above to prevent overshadowing of the original query.

This trigger-pair model itself refines the result from google to a large extent but this still needs to be ranked based upon the user's interest stored in his profile. To do it, PIWA first gets the top 20 results from google, then downloads each of the 20 pages and generates keywords using the keyword generation algorithm explained above for each of the 20 pages. Then a score for each page is computed by summing up the product of weights of each matching word in the matched interest and the keyword generated for the page. The pages are then ranked on this score. The resultant top 10 urls of the pages form one of the results. Another result of top 10 urls is generated which gives half weight to the rank given by PIWA and half weight to Google. Both the results are available to the user.

Chapter 4

The Agent Architecture

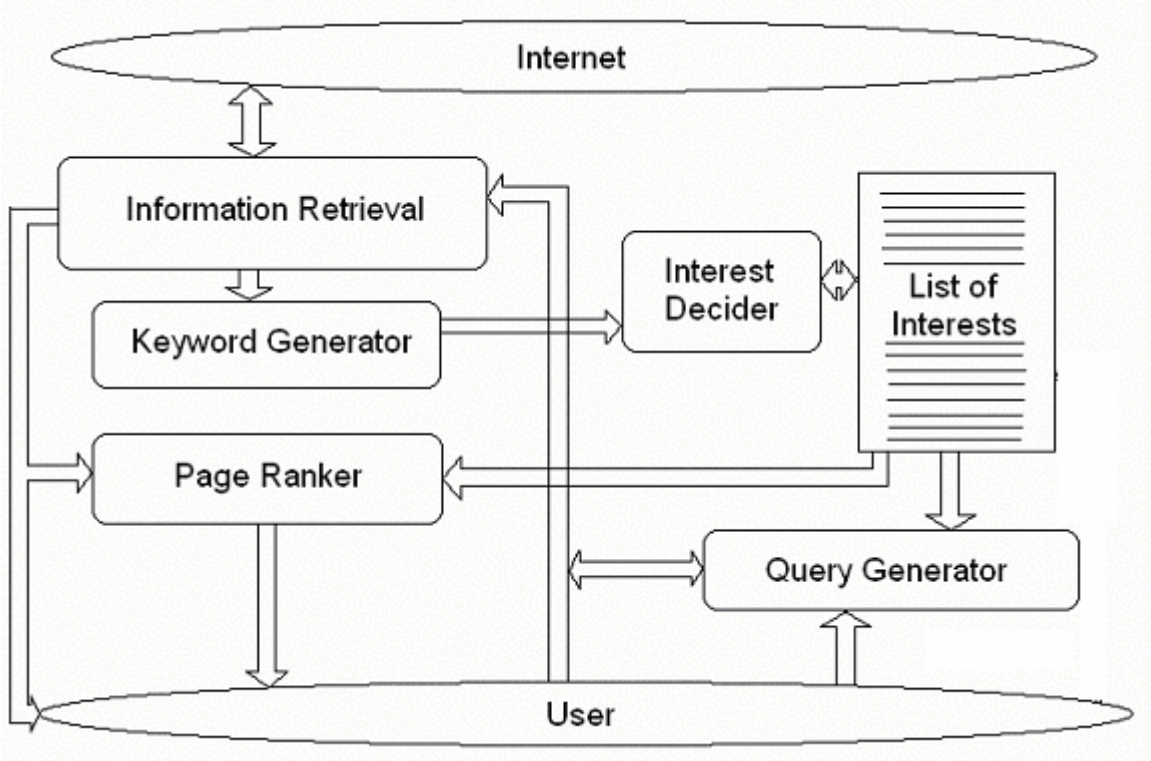

Fig 2: The proposed agent architecture

The architecture of the Personalized Intelligent Web Agent is shown in the above diagram. The *user* and the *internet* form two boundaries for the *agent*. The user browses the net in his usual way. The *information retrieval* module forms the proxy and accesses requested pages from the net and forwards them to the *user* (browser) and also for the *keyword generator* module. The *keyword generator* module generated keywords representing the page and passes the keywords to *interest decider* module which decides whether the keywords form a new interest or should be merged with an already formed interest. The *list of interests* is hence updated based on the decision. Now when the user searches for some particular query or similar page, the appropriate query is generated by the *query generator* module and resultant query passed on to *information retrieval* module to get the results from *google*. The results are passed on to *page ranker* module which ranks the pages based on the interests of the user or the newly created interest based on the query. The result is displayed to the user.

Chapter 5

Implementation Details

5.1 System Configuration

The PIWA was developed on Java (version 1.4+) hence is platform independent. It is tested on a Pentium III, 128MB RAM machine and performs well. Proxy support in provided for user browsing but to use the search facility an open net connection is required.

5.2 Knowledge Representation

The PIWA stores knowledge in a two-tier system. The user profile consists of interests and interests consist of keywords having weights and a timestamp. A keyword with its weight forms the element of this knowledge base and is represented as:

```
class element
{
      public String word;
      public int val;
      public void element()
      {
            word =new String("");
            val = 0;
      }
}
```

Thus each element has a word and a value which are initiated to a null string and zero respectively.

10 such elements followed by a special element having its word as “1234567890” and value as the timestamp of the interest form an **Interest.**

The profile of an user consists of a maximum of 20 such interests.

Another element names gelement is used specially for the google results.

```
class gelement
{
      public String title;
      public String snippet;
      public String url;
      public void gelement()
      {
            title = new String("");
            snippet = new String("");
            url = new String("");
      }
```

}
This special element stores the title, snippet and url of the results returned by google.

5.3 File Summary

1. WebAgent.java – This is the initiative program for the PIWA. It is responsible for kickstartng the WebAgent. It calls various other classes and gets the agent to work.

2. newThread.java - It acts as the proxy server for the user requests. It is configured to listen at port 3128 to which the browser's proxy in configured. On receiving request from the browser, it is responsible to download the page and keep a copy of it as cache. It also checks whether the request is black listed or not and if it is the user is prevented to view it.

3. ComboBoxDemo.java – The only graphical user interface for the user which is kept simple to be as unobtrusive to the user as possible. It requests for the proxy configuration and thereby keeps silent keeping a track of the web pages visited by the user.

4. ParserHTML.java – On receiving a html file, it parses it, find the various test embedded in HTML tags, gives them weights, sorts them out and thereby computes the top 10 keywords for the page which it writes out to the "keywords.dat" file.

5. threadInterest.java – It is responsible to check whether the keywords generated match with any of the interest already present in the list of interest or is a new one. In the former case, it updates it while in the latter case it appends the new group of keywords into the list of interests.

6. interest.java – It provides routines to read and write into the list of interests and also maintains the list of interests.

7. test.java – It allows updation of list of interests from already downloaded web pages.

8. testsearch.java – It allows the use of search facility by receiving the user query as its command line argument.

9. interest.dat – The list of interest.

10. keywords.dat – The list of keywords generated for a page.

11. blackListURL.exc – The list of black-listed urls.

12. googleapi.jar – The google api is needed to do a google search.

Chapter 6

Results

6.1 PIWA features

1. The PIWA generates a quite accurate list of interests which can be used in many applications besides search.
2. The PIWA uses the list only to implement web searches for a given user query.

6.2 Case Studies

6.2.1 IRAQ

A user interested in current news in Iraq (Dt. April 14, 2004) goes to news.google.com and news.yahoo.com and surfs 10 pages on it. The list of pages are:-

1. The Daily Star - Opinion Articles - Concorde, Lexington, Fallujah and Najaf: Why folks fight (http://www.dailystar.com.lb/article.asp?edition_id=10&categ_id=5&article_id=2054)
2. Top News Article | Reuters.com (http://www.reuters.com/newsArticle.jhtml?type=topNews&storyID=4822366&section=news)
3. Aljazeera.Net - Occupation prepares siege on holy city (http://english.aljazeera.net/NR/exeres/B6C252A9-AF36-43E7-A6AD-EE7CB86532A1.htm)
4. Boston.com / News / World / Middle East / Tension, talks at a holy city (http://www.boston.com/news/world/middleeast/articles/2004/04/14/tension_talks_at_a_holy_city/)
5. TheStar.com - U.S. set to `bite rattlesnake' (http://www.thestar.com/NASApp/cs/ContentServer?pagename=thestar/Layout/Article_Type1&c=Article&cid=1081894211527&call_pageid=968332188854&col=968350060724)
6. Mercury News: Bay Area news, technology, jobs, cars & real estate (http://www.mercurynews.com/mld/mercurynews/news/world/8427437.htm?1c)
7. Cleric bends, but US hardens stand - The Times of India (http://timesofindia.indiatimes.com/articleshow/616942.cms)
8. Calgary Herald - canada.com network (http://www.canada.com/calgary/calgaryherald/news/story.html?id=3968ac2a-9ec2-4ac3-8a0d-2d0f6deb4cce)
9. Khaleej Times Online (http://www.khaleejtimes.com/DisplayArticle.asp?xfile=data/focusoniraq/2004/April/focusoniraq_April126.xml&section=focusoniraq)

10. Reuters | Latest Financial News / Full News Coverage (http://www.reuters.co.uk/newsPackageArticle.jhtml?type=topNews&storyID=492986&section=news)

After browsing through 10 pages, a total of 7 interests are formed which are:

iraq 17 fallujah 17 daily 17 star 16 lexington 13 concorde 13 fight 12 folks 12 najaf 12 opinion 12 1234567890 1	document 94 news 78 write 56 reuters 35 iraq 32 said 17 ns4 16 navigator 14 useragent 14 indexof 14 1234567890 10	said 42 news 39 iraq 37 sadr 30 city 28 document 24 holy 23 reuters 20 occupation 17 aljazeera 16 1234567890 3	thestar 14 rattlesnake 12 bite 12 set 12 story 8 iraq 7 star 6 tacoda_ams_ddc_js 5 canadian 5 said 5 1234567890 5
india 3 times 3 indiatimes 3 blockerror 2 search 2 documents 2 true 1 onerror 1 window 1 timesofindia 1 1234567890 7	calgary 17 herald 16 sadr 11 canada 11 network 10 iraq 4 cleric 4 canwest 4 2004 4 subscribers 4 1234567890 8	iraq 42 fallujah 29 forces 28 marines 24 iraqi 20 said 20 baghdad 20 two 16 coalition 16 tuesday 16 1234567890 9	

Only 7 interests are formed as for 3 pages, the interests have merged with previously formed interests. Take the example of first interest formed from the article in Daily Star. The top 4 keywords are *iraq, fallujah, daily and star* which is very much related to the news. Then comes *Lexington* and *concorde* which entered into our list as they were part of advertisements and other news articles in the same page. Our parser does not distinguish between main news items and secondary ones as all lie in the same page. A better parser to eliminate this extra matter can be developed but that requires idea about the structure of the page which we can not have in our case. Then comes *fight, folks, najaf and opinion* which are very much a part of the page. At the end we have our separator as well as the timestamp i.e. 1 showing it is the first page visited. It can be noted that for the second the timestamp is 10 showing the merging of the 2nd page with the 10th page (both are from reuters).

Now we try to search for "iraq". First of all it searches for matched interests and it finds a match with six interests of which the last one has the highest weight that is 42. Hence a

single word fallujah is appended to the query and now a google search for "iraq fallujah" in invoked. Now we need to compare between the three results namely that from Google, from PIWA and PIWA + Google mixed.

In direct google results for "iraq" not a single match is found in top 10 results which relate to the incidents at fallujah (in which the user is currently interested). So better try to invoke a google search on "iraq fallujah" to show how google results are ranked again to user's interests. So the results are:

Google	PIWA	PIWA + Google
Fallujah -1	10	5
Fallujah / Habbaniyah - Iraq Special Weapons Facilities – 2	5	1
Fallujah Iraq News: Iraq Fallujah - 3	4	2
Fallujah Iraq News: Chinese Group Abducted in Iraq; Fallujah Truce ... – 4	3	3
SF Examiner: Looking for stability in Iraq, Fallujah – 5	8	6
Fitness guru among 4 killed in Iraq (FALLUJAH Outrage) – 6	9	8
Frugal's World of Simulations - Iraq - Fallujah - burned bodies of ... – 7	1	4
The Spokesman-Review.com - Fighting spread in Iraq; Fallujah ... -8	7	9
CNEWS - World - Iraq: Fallujah residents bury their dead, try to ...	6	10
sacbee.com -- Iraq -- Fallujah fighting claims more than 280 ... – 10	2	7

Thus we see that the first result by google is a description of fallujah and has less to do with the fighting going on there so it rank a poor 10 in PIWA ranks whereas the seventh result of google which is basically from a discussion board with a lot of talking going not only about iraq and fallujah but also marines, forces, coalation, Baghdad (words in the matched interest) ranks first in the PIWA ranking with a comfortable lead of scores over the second result (5309 vs 2759). The mixed results tends to moderate both the extremities and results in good results. Its ranks 1 is 2nd of google and 5th of PIWA.

After this lets move to another simpler example:

6.2.2 Sandeep

Took just 3 pages

1. Sandeep Kumar (www.geocities.com/sandyiitg/)
2. Studs - Sandeep Kumar (www.iitg.ernet.in/studs/Members/sandeep)
3. Sandeep Kumar (www.geocities.com/sandyiitg/resume.html)

A total of 3 interests were formed.

kumar 11	studs 19	resume 13
sandeep 11	sandeep 15	developed 9
some 4	kumar 11	data 8
tawang 4	2004 6	part 7
view 3	iitg 6	software 7
indian 3	yahoo 5	iitg 7
site 2	iit 5	net 6
old 2	home 5	systems 6
search 2	browser 4	system 5
guestbook 2	design 4	project 5
1234567890 1	1234567890 2	1234567890 3

The interests are formed on expected lines. Now on searching for "sandeep" in google we get quite out of context results so we compare the three results for the query "Sandeep kumar" which is triggered from the second interest.

Google	PIWA	PIWA + Google
About Sandeep Kumar Shukla (Sandeep) – 1	4	1
Sandeep Kumar – 2	> 10	8
Links – 3	> 10	> 10
Dr. Sandeep Kumar's home page – 4	> 10	> 10
DBLP: Sandeep Kumar – 5	7	3
DBLP: Sandeep Kumar Goel – 6	3	2
RE: Bindings from Sandeep Kumar on 2002-03-21 (www-ws-desc@w3.org ... – 7	5	4
RE: Requirements: Describing WS Capabilities from Sandeep Kumar on ... – 8	6	5
Index of /pub/papers/sandeep-kumar – 9	10	9
Studs - Sandeep Kumar – 10	2	5

This is classic example where a past visited page can change the rankings greatly. The page visited was "Studs - Sandeep Kumar" which is page number two and also the matching interest. It ranked 10^{th} in Google but 2^{nd} in PIWA ranking giving a final of 5^{th} rank on PIWA + Google. Thus a page visited in the past forming a matching interest alters rankings greatly. As the query "Sandeep" was too general a word still results close to the visited pages were pushed up. If the query is modified to "Sandeep Kumar" then the final query submitted to google in "Sandeep kumar 2004 iitg". Comparing among them.

Google	PIWA	PIWA + Google
Studs - Sandeep Kumar – 1	1	1
B. Tech 00 - Department of CSE, IIT Guwahati – 2	3	2
Sandeep Kumar (pdf) - 3	>10	>10
Sandeep Kumar – 4	6	4
Alumni IITG – 5	> 10	8
Studs - BTech Cse 2000 – 6	4	5
Studs - Btech ECE 2001 – 7	2	3
January 2004 – 8	7	6
Teams - Public Information – 9	5	6
Teams - Public Information -10	10	10

Here we see that 3rd result of google is a pdf which can not be parsed by our parser hence does not figure in our list. Else results are on expected lines.

Another example is the

6.2.3 India Pakistan Test Series

The pages considered are:

1. India looking solid (http://inhome.rediff.com/cricket/2004/apr/14india.htm)
2. Sporting Life – Cricket (http://www.sportinglife.com/cricket/news/story_get.dor?STORY_NAME=cricket/04/04/14/CRICKET_Pakistan_Lead.html)
3. Reuters | Latest Financial News / Full News Coverage (http://www.reuters.co.uk/newsArticle.jhtml?type=cricketNews&storyID=4823063&section=news)
4. Daily Times - Site Edition (http://www.dailytimes.com.pk/default.asp?page=story_12-4-2004_pg2_3)
5. cricmania - News: Cricket-India 203-3 v Pakistan (224) - third test, tea (http://www.cricmania.com/cricket/reuNews/index/user/us03/ref/2004-04-14T094456Z_01_B667998_RTRIDST_0_SPORT-CRICKET-INDIA-TEA.html)

As the pages were very similar in their content, only two interests were formed

pakistan 64	news 26
india 63	document 24
test 61	reuters 17
cricket 50	indexof 14
april 40	useragent 14
2004 23	navigator 14
england 22	write 13
fourth 22	full 11
reuters 21	india 10
tea 20	coverage 10
1234567890 5	1234567890 3

Now if we search "India pakistan". Google by default results in all war related sites but this is not what the user seeks as he at present interested in the test series going on. So using trigger pair the resultant query is "India Pakistan test cricket". Comparing the results:

Google	PIWA	PIWA + Google
Wisden CricInfo – 1	> 10	7
India Pakistan Test Cricket Pack : Indiaplaza.com DVDs and Movies! - 2	8	3
BBC SPORT | Cricket | Sehwag pummels Pakistan – 3	5	2
BBC SPORT | Cricket | Pak v Ind | Pakistan v India: As it happened – 4	2	1
Live Cricket Coverage – 5	> 10	8
Reuters AlertNet - India, Pakistan face off for final cricket test – 6	7	4
FOX SPORTS | Cricket ->Cricket Records Home – 7	> 10	> 10
Ganguly out of historic Pakistan Test - Cricket – www.theage.com. ... – 8	> 10	10
Pakistan V India - 3rd Test - Cricket Betting	>10	>10
123India.com Cricket – 10	> 10	> 10
cricmania - News: Cricket-India 107-4 v Pakistan - second test ... – 14	4	5
cricmania - News: Cricket-India 234-7 v Pakistan - second test ... – 15	3	6

Here there is some great disparity in Google and PIWA results. Like the 7^{th} , 8^{th} , 9^{th} and 10^{th} result of Google cant make it to top 10 of PIWA whereas 14^{th} and 15^{th} ranked news were ranked healthily at 4^{th} and 3^{rd} rank as they were much closer to the matched interest.

So from the above results we can conclusively say that the results of PIWA are more relevant to the user as it matched to the user's interest well. The PIWA + Google results are best to use for any average user if he does not want to be quite constrained by his past interests. In classical AI term, the PIWA results are based completely on "exploit" strategy whereas the PIWA + Google results have an explorative edge.

6.3 Failures:

1. The graphical user interface or some web based interface for the search tool is needed for better user interaction.
2. Features like ability to handle plugins are needed so are to parse through docs, pdfs, etc
3. Advanced plugins are required which support SSL encryption, zipped data so as to parse encoded pages. A typical example is that of www.google.com which

returns its index.html as gzip data which requires plugins to unzip and hence parse which the present PIWA cant do.

6.4 Feedback:

The Agent application was tested with real users and some of the positive and negative feedbacks are:

1. The keyword generation and profile management modules work very good.
2. Additional user controls should be provided or user interface improved to support user determining number of words in interest, number of interests, etc.
3. A Gui for search tool is required for any practical usage of the search utility.
4. The search by PIWA seems to constrain the search too much but PIWA + Google works fine.

Chapter 7

Conclusions

The current Project on "An Intelligent Web Agent" for web searches imposed two daunting challenges - Efficient profile generation and no active user feedback. Efforts were made to build an assisting web agent by capturing user's browsing habits. Machine-learning techniques were used to manage and learn the user profiles and keep them updated with the passage of time. An Interface to the agent was provided to allow the user to control the agent application. The observed results were encouraging

7.1 Extensions and Future Work

The support of plugins can be added as well as a GUI for the search tool made. A better user interface for the whole agent can be made giving the user more power to configure the agent.

A future extension to it can be a personalized news agent which can provide news items from various online news sites. In this case special parsing techniques can be invoked for special site but this will constrain the domain of search.

References

[1] Jeffrey M.Bradshaw, “Software Agents”, MIT Press 1997, pp.12-17, 296-297.

[2] Joseph P.Bigus and Jennifer Bigus, “Constructing Intelligent Agents using Java”, Wiely-2e, 2001, pp: 9-10.

[3] Russell, Stuart, and Peter Norvig. Artificial Intelligence A Modern Approach. Prentice Hall, Upper Saddle River, N.J., 1995, pp 31.

[4] T. Kuflik, P. Shoval, “Generation of user profiles for information filtering”, Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval, Athens, Greece, pp: 313 - 315 , 2000.

[5] Kjersti Aas, “A Survey on Personalized Information Filtering Systems for the World Wide Web”, Report no. 992, Norwegian Computing Center, December 1997.
http://www.nr.no/~kjersti/agent.html

[6] T. Joachims, D. Freitag and T. Mitchell. WebWatcher: *“A tour guide for the World Wide Web*”, *"Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence ({IJCAI}-97)"*, 1997.
http://citeseer.nj.nec.com/joachims96webwatcher.html

[7] Liren Chen and Katia Sycara, “ *A Personal Agent for Browsing andSearching”, "Proceedings of the 2nd International Conference on Autonomous Agents”,* 1998.
http://citeseer.nj.nec.com/chen98webmate.html

[8] Henry Lieberman, "Letizia: An Agent That Assists Web Browsing", "Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence ({IJCAI}-95)", Morgan Kaufmann publishers Inc.: San Mateo, CA, USA, 1995.
http://citeseer.nj.nec.com/lieberman95letizia.html

[9] M. Pazzani, J. Muramatsu and D. Billsus, "Syskill & Webert: Identifying interesting Web Sites," In Proceedings of the Thirteenth National Conference on Artificial Intelligence, pp. 54-61, Portland, 1996.
http://citeseer.nj.nec.com/pazzani98syskill.html

[10] Rob Barrett, Paul P. Maglio and Daniel C. Kellem: “WBI: How to Personalize the Web”, In Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI '97), Atlanta, GA.
http://wwwcssrv.almaden.ibm.com/wbi/papers/chi97/wbipaper.html

[11] G. L. Somlo and A. E. Howe, “Using web helper agent profiles in query generation”, Proceedings of the second international joint conference on Autonomous agents and multiagent systems, Melbourne, Australia, pp: 812 - 818, 2003. http://www.cs.colostate.edu/~howe/papers/aamas03.pdf

Contact Information:-

Name: Sandeep Kumar

Permanent Address:
C/o – Mr. P. R. Mahato
Qr. No. – 2072,
Sector – 4/D,
Bokaro Steel City,
Bokaro,
Jharkhand.
Pin – 827004
Ph. No. – 06542226171

Email id:
sandyiitg@yahoo.com
sandyiit@postmark.net