Distributed User Interest Modeling

And

WWW Reference Locator

By

Paul Card

A Thesis
Submitted to the Faculty of Graduate Studies
In Partial Fulfillment of the Requirements
For the Degree of

MASTER OF SCIENCE

Department of Electrical and Computer Engineering
University of Manitoba
Winnipeg, Manitoba

THE UNIVERSITY OF MANITOBA

FACULTY OF GRADUATE STUDIES

*****

COPYRIGHT PERMISSION PAGE

Distributed User Interest Modeling and WWW Reference Locator

BY

Paul Card

A Thesis/Practicum submitted to the Faculty of Graduate Studies of The University of Manitoba in partial fulfillment of the requirements of the degree of Master of Science

0.1 Abstract

The Internet is the largest interconnection of computer networks ever built. One of its strengths and weaknesses is that it is distributed across the world. The World Wide Web (WWW) specifically is the largest data repository ever built and continues to grow at incredible rates [5]. People from all over the world are constantly revising, contributing, and viewing its content. One of the problems of such a large distributed data base is trying to locate and access specific information.

Like a library the web needs a form of index or pointers that can direct the users to the specific information they require. One of the techniques that has been used are search engines. Search engines attempt to build an index of web pages by crawling the Internet one site at a time. This is obviously a linear approach and falls in to a O(N) time complexity where N is the number of web pages on the Internet. This would work fine if the Internet was a constant size and the existing content unchanging. In time, a large enough web crawler would visit all of the sites on the Internet and be able to create an accurate index. Because the Internet is constantly being revised and its user base growing the search engine technique is failing. Even the

largest of search engines, AltaVista has only 16% coverage of the Internet and dropping [5].

The web has data and users spread all across the world. Because there is no notion of physical distance on the web, users are just as likely to visit web sites hosted locally as they are to visit sites across the globe. This creates a large amount of traffic for backbone networks. One proposed technique to help reduce the traffic on the Internet is web caching. Web caching attempts to keep copies of popular content in a local cache. This should save the cost of a later download by the same or another user. Studies in world wide web (WWW) request patterns have indicated that 70% of requests from a web caching group are one time references [20]. This implies that in a group of people there is very little overlap in the content they view.

This thesis describes a system that helps increase the overlap of the pages viewed by a local cache group. It works by passive learning what each user is interested in and builds a small database of the sites that they find to be good references. It then shares these databases with other users with similar interests. This shows promise of becoming a much faster and more accurate way of finding references compared to search engines. It also has

application in e-commerce as on-line businesses are very interested in their visitors interests.

0.2 Acknowledgments

I would like to extend my gratitude and thanks to Dr. McLeod, **TR***Labs* and the Department of Electrical and Computer Engineering.

Contents

Chapter 1

Introduction

1.1 Motivation

The Internet's World Wide Web (WWW) is the largest resource of information ever created. It is also growing at near exponential rates [5]. This is both a blessing and a curse to the end user. The wealth of information available is becoming more and more widespread and obtainable. Unfortunately, the sheer volume of information available on the WWW is unmanageable. Current techniques for indexing the WWW for quick retrieval are starting to show their inherent weaknesses. The true shortcoming of the typical search engine technique is its centralized approach. As the Internet continues to

grow at an exponential rate, a centralized search engine would not keep up without an exponential growth of hardware and bandwidth to the engine. In order to build a scalable index into the WWW, a system that scales with the growth of the Internet is required.

One such area that has seen wide spread publicity is Mpeg3 encoded music or MP3s. MP3s are typically in the order of 3-5Mb each and encode one 3-5 minute song. The original approach for the distribution of these files was through the traditional client server model. Unfortunately due to the large size of MP3s and the limited Internet bandwidth downloading MP3 files proved to be quite slow. In order to address this problem, a company called Napster [23] developed a distributed peer to peer database system that allowed people to exploit a more scalable approach of a distributed database. By distributing the database it becomes much easier to exploit data that is "closer", in network cost and bandwidth to the end user. Napster works by leaving the data on the end clients. The Napster service is a centralized directory of the content that each client has to publish. If the same techniques are utilized to build an Internet search engine, then the scalability problems would be resolved.

Due to legal troubles [24] Napster's centralized directory has been shown to be a weakness to the model. The central server is a single point of failure that can cause the whole system to become inaccessible. Gnutella [27] is an open source Napster like program, that attempts to rectify the centralized server weakness. It discovers peers on the network without use of a centralized failure point. Unfortunately, performance suffers. Later on we will describe an architecture that falls between the Naspter and Gnutella models.

Studies have shown that the majority of WWW requests are one time requests [20]. This means that the majority of the data being requested is never being revisited. This is a problem for web caching techniques and is generally inefficient. The proposed solution is to have an agent assigned to each user. This agent learns its user's interests by observing its users WWW requests. At the same time the agent builds a database of the sites most popular with its assigned user. Then by discovering other agents on the network with similar interests the agent can trade references. By making recommendations of sites visited by people locally the one time hit problem is reduced as a local cache may already have these pages stored in them.

Chapter 2

Related Research

2.1 Web Collaborative Filtering

Collaborative filtering is a term coined by Professor, Pattie Maes of the intelligent agent research group at the MIT media lab [11]. Collaborative filtering is technique used to give agents the ability to make suggestions to humans in domains that have a high degree of personal taste. Collaborative filtering leverages the notion that the system need not understand a data item, only that it is interesting to the user [12]. As such the agent need only to be able to recognize the fact that two data items are similar. Given that the agent can find clusters of data items, it need only identify which clusters

are of interest to its owner. Collaborative filtering has been shown to be quite effective in a number of domains, music selection [6], movie critique [8] and news [9]. Currently the architecture for WWW collaborative filtering does not exploit the distributed multi-agent features of agent technologies that is inherent in this problem domain. One of the problems of these techniques and their implementations is that they require a large centralized database of user interests. This, like the Internet search engine centralization problem, suffers from scalability problems.

2.2 Assistant Agents

User assistant agents has been suggested as a better model for user modeling. The traditional architecture associates an intelligent learning agent with each user. An example is MIT's Web Hound [10]. As the user requests pages on the Internet, the agent also parses the pages to generate an averaged user interest representation. This part of the system can reside either on the users machine or at some central web access point like a firewall.

2.3 Active Filtering Agents

Once an agent has identified its user's interests, it must seek out additional web pages of interest to the user. An active filtering agent acts much like a web crawler searching out additional pages. The agent will start with several pages of interest to its user and perform a parallel tree traversal looking for similar pages of interest to their user. Unfortunately, this approach generates more traffic then a user searching on his own (MIT's Web Mate) [13]. As well, the reference sites generated are often ones that the user would have found shortly.

2.4 Passive Filtering Agents

A simple extension to active filtering is to treat the local Internet connection as a content stream like Internet news [9]. Passive filtering attempts to solve the active filtering agent problem by placing itself in a position in the network where alot of Internet traffic is present, usually at an Internet access point. At this access point a large number of other users requesting data that the agent can passively monitor for pages of interest to its user and then forward references to him. This does solve the increased bandwidth

problem, but creates another problem. If all the users using this access node have agents filtering on their behalf then all pages being requested need to be processed by everyone's agent. There are techniques to deal with these issues such as cache and copying pages as they pass but this unfortunately, only masks the scalability limitations of the solution.

2.5 User Modeling

The key to web filtering is the ability to take a set of web pages and generate an interest representation. This applies to both cooperative and non-cooperative agents. A users interest can be represented as a subset of topics taken from the universal set of all topics discussed on the Internet. As the agent only has access to the pages viewed by its user, it really only needs to identify the topics of interest taken from the set of topics viewed in the user's down-load set. Unfortunately, in order to compare pages by topic, the topic of each page needs to be extracted to some machine usable format. This is typically done by using information retrieval techniques based on statistics of large data sets. Term Frequency Inverse Document Frequency (TF/IDF) is used in this thesis to identify which words to use as topics.

Once a dictionary of topic words is created a topic representation can be created for each page by forming a vector of existence levels (percentage of times a word appears in the page) for each word in the dictionary [4]. From there Unsupervised Neural Networks can be used to successfully cluster the page representations by topic [14]. This is done to identify the pages that are "close" in the topic dimensional space or in other words discuss the same topic.

An additional piece of information that is rarely used, is the level of user interest in each page. As the user views various pages, the agent not only categorizes the pages viewed by topic but also learns an interest scalar for each page viewed. This can be done by looking at a number of simple parameters. The first and simplest parameter is whether or not the user placed the page into a bookmark file. Web users typically bookmark pages on the Web for quick retrieval during a later session, therefore it must be of relatively high interest to them. Another parameter used is the number of times a page is revisited. If a user finds a page interesting or a good reference, it will be revisited frequently. The last parameter is the inter-request time between pages. The inter-request time is defined as the time between the user selects

a link to arrive at the current page and time when the user selects a link to leave the current page. A long inter-request time would normally indicate a high degree of interest in the page. During our work we found that on a number of occasions users would open a page then leave their computers. This creates an artificially high interest in a page that was not interesting to the user. To address this we use the inter-request time to lower the interest parameter. This is done by lowering the interest parameter on pages where the time a page is viewed is under 5 seconds. Pages that are quickly passed over by the user are considered to not be of interest. These are all useful pieces of information in determining the level of interest in each of the pages. The interest measure of pages also serves as a quality measure for each of the pages. This can be useful when suggesting a page to another person as it gives a human rating to the page quality [18].

2.6 Overview

In order for the agent communication and user interest learning to be efficient, web pages must first be rendered into a form conducive to computer manipulation and communication. Chapter 3 discusses the formation of a

normalized representation of a web page. It also includes the discussion of algorithmic consideration and modifications made to standard information retrieval techniques to provide real-time rendering. It also discusses the need for and steps taken to generate a topic word dictionary necessary for efficient rendering.

Once a set of normalized web pages is generated, it needs to be clustered together in order to extract the cluster centers. The cluster centers represent the broad interests of each user. The process of keeping a current interest database is discussed in Chapter 4. As well, the development and modifications to the algorithms used.

Chapter 5 looks into the challenges of creating a universal dictionary that represents all topics discussed on the Internet. It also discusses the feasibility of each technique. In Chapter 6 the technique used for this thesis work is described, as well as some of the challenges encountered.

With the ability to identify user interests, a multi-agent system is developed to aid in the discovery of relevant web pages. By leveraging the clustering ability of each user's agent, references are simply represented by a

topic vector and a Uniform Resource Locator (URL). The development of a multi-agent architecture and communication system is developed in Chapter 7. Chapter 8 documents some results found during the work.

Chapter 3

Web Page Representation

3.1 Introduction

In order for the agent communication and user interest learning to be efficient and effective, web pages must first be rendered into a form agreeable to computer manipulation and communication. In addition, the learning process is done using Artificial Neural Networks (ANNs). These techniques require that some amount of preprocessing is done before the algorithms become effective [15].

3.2 Web Page Parsing

The first assumption made here is that the textual content of a web page can be used to indicate the topic the page discusses. This is usually a safe assumption to make, but may fail in some circumstances. Much of the information may be contained in other formats. Most modern web browsers support many file and media formats including images, sound and video. Some work has even been done trying to extract the information contained in the structure of the page itself [14]. In this work we assume that the topic content can be extracted from the text alone. A further addition would be look into other data formats. In the case of wireless applications using protocols such as Wireless Access Protocol (WAP) the text assumption is well justified.

Figure 3.1: Raw HTML Page Source

As a simple example... Consider this web page

```
<HTML><h1>Laptops</H1></BR>
Laptops are a portable format of modern computers.
They are light weight and are often used
for <tt>business</tt> and people on the go.
They have many of the same feature as their
desktop counter part.  Such as fast <b>CPU's
800-1000MHz</b> and Hard Disk space in the order
of many<b>Giga-bytes</b>.<HTML>
```

3.3 Tag Removal

The first step in the parsing is to remove the HTML tags that are used for formatting the text from the actual textual content Figure 3.1. The tags are delimited by "<" and ">" and can be nested [26]. The removal of the tags can be done with a quick single pass through the document. Typical output illustrated in Figure 3.2.

Figure 3.2: Source with HTML Tags Removed

To continue the example... With the HTML tags removed

```
Laptops Laptops are a portable format of modern
computers They are light weight and are often used
for business and people on the go They have many
of the same feature as their desktop counter part
Such as fast CPUs 800-1000MHz and Hard Disk save
in the order of many Giga-bytes
```

Figure 3.3: Source with Stemmed Words

The same text after Porter Stemming

```
laptop laptop portable format modern comput light
weight often busi peopl same featur desktop
counterpart such fast cpu mhz hard disk spave
order mani gigabyt
```

3.4 Text Normalization

The next step is to normalize all of the words into their root format. This eliminates the variation in conjugation of words and brings plurals together. The algorithm used is the Porter Stemming Algorithm, a standard stemming technique for removing most of the "morphological and inflectional endings from words in English" [21] and has been used in related work [14]. Figure 3.3 illustrates the output of the stemming process.

3.5 Topic Word Extraction

At this point the words and their frequencies are filtered using a pre-computed topic dictionary. The dictionary contains all of the terms that are currently being considered as potential topic words. These words are said to have a high level of salience. Their selection is discussed later. Any word that does not appear in the dictionary is added to an inactive state. This allows statistics of their appearance frequency to be collected. These words are stored as they may be used for later revisions of the dictionary. During this process, common stop words are removed. Stop words are words considered to be too common to be of any value in identifying the topic. As well, words that are very infrequent are removed as they do not represent a more general topic. These are pulled from the document early on to help reduce processing overhead. There are a number of standard stop word lists available. The one used in this thesis work is included as an appendix [25]. These lists are captured by their exclusion from the topic dictionary.

Figure 3.4: Resulting Web Page Representation

Here we would have a total word count for the document and a vector of indices

```
Total # Words:[(Word Index, # of Appearances)...]
```

3.6 Final Page Representation

At this point, a single web page is represented by a count of the total number of terms in the document and a vector of indices into the topic dictionary as shown in Figure 3.4. This is done in order to save a small amount of memory rather than storing the topic terms multiple times. In addition, each entry in the vector contains the number of times the term appeared in the document. The number of times a word appears in a document is divided by the number of words in the document to represent a point in that particular word's topic dimension. This makes every document a point in the space defined by the topic dictionary. In addition, these statistics are used later on

in dictionary generation. The next step is to integrate the many web pages visited into a more general representation of topic interests. In addition, the HTML source is stored in case it needs to be reclustered using an updated dictionary.

Chapter 4

Web page clustering

4.1 Introduction

In monitoring the download patterns of a user, a data set of web pages is accumulated. From the techniques described previously, we can form a data set of points in topic space. We use this distribution as a representation of an over-specific user interest set. The problem now becomes a matter of generating a more generalized representation of the topics contained in those pages. In other words we want to extract the users core interests from the sparsely populated topic space.

4.2 User Interest Level

An additional piece of information that is rarely used is the level of user interest. As the user views various pages, the agent not only categorizes the pages viewed by topic but also learns an interest scalar for each page viewed. This can be done by looking at a number of simple parameters.

- R Revisit Count - The number of times this site has been visited in the current DB
- T Time Viewed - The amount of time the page was viewed in seconds
- B Book-marked - A boolean indicating whether or not this page is in the book-mark file

The first and simplest parameter is whether or not the user placed the page into a book-mark file. Web users typically book-mark pages on the Internet for quick retrieval during a later session, therefore it must be of a relatively high interest level to them. Another parameter is the number of times a page is revisited. If a user finds a page interesting or a good reference it will be revisited frequently. The last parameter is inter-request time between pages. Normally a long inter-request time would indicate a high level

Figure 4.1:

$$S = \begin{cases} T < 5 & \frac{T}{5} \\ T >= 5 & 1 \end{cases} \times \begin{cases} BookmarkedPage = True & 1 \\ BookmarkedPage = False & \frac{R}{TotalVisitsinDB} \end{cases} \quad (4.1)$$

of interest in a page. Unfortunately we account for the case when a user leaves for a coffee and doesn't return for several minutes. Because of this we use the inter-request time not as an additive measure but a subtractive measure. If a user views a page for a very short time <= 5 seconds we subtract from the user interest level parameter. These are all useful in determining the level of interest in each of these pages. The level of interest also serves as a quality measure for each of the pages. This quality measure can be useful when suggesting this page to another person as it gives a human rating to the page quality [18]. The quality measure is assigned using the equation 4.1. For example if a user viewed a page for 3 seconds and the page was bookmarked, S would evaluated to $\frac{3}{5}$.

4.3 Data Structure

As the user views pages on the WWW, new points in the topic space are added to the set. As well we have a measure of the user interest S for each page. We would like to build a data structure that allows us to add and remove these points and adjust their interest levels from the space defined by the topic dictionary. From this we would also like to be able to query a small set of points from the structure that indicate the user's current interests. Another feature required is the ability to remove data points. The data points in the structure also need to be aged off as the user's interests decrease. This is done by associating a TTL (time to live) parameter with each point. The TTL ensures that the structure represents the user's current interests, limits the size of the DB and ensures that stale page references are not being suggested to other users.

The data structure is based on an ANN (Artificial Neural Network) technique called ART that clusters points. The input are the points in topic space. The results are the cluster centers from the ANN assuming that the topic space is correctly formed. The cluster centers represent the center of an averaged topic region that the user has visited many pages in. This region

is assumed to represent one of the user's current interests.

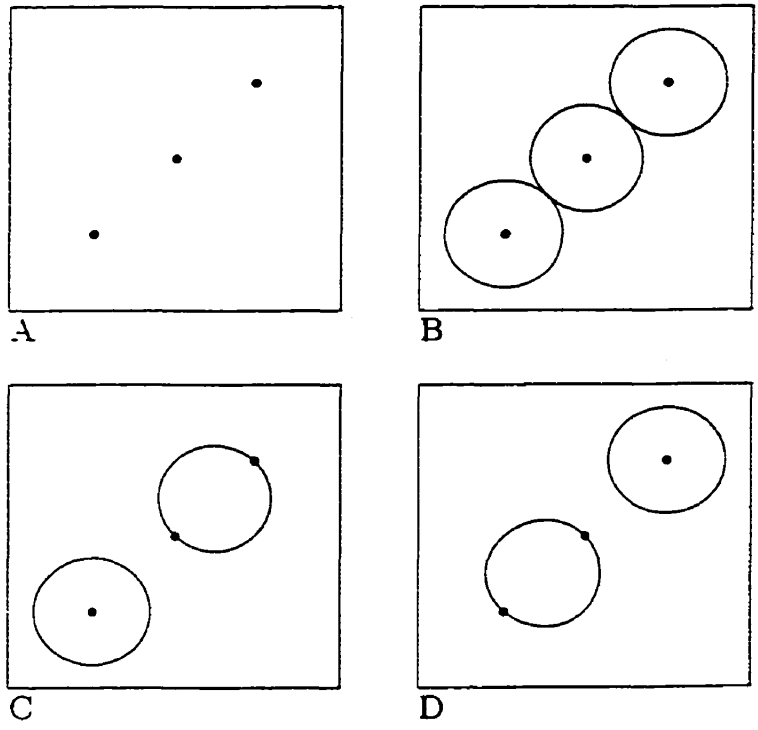
4.4 ART

Adaptive Resonance Theory (ART) is an unsupervised learning technique developed by Carpenter and Grossberg [22]. The technique is being used here to cluster the individual page points in topic space. If the topic space (topic dictionary) provides appropriate resolution in the required areas, the clusters will represent the interests of the user.

The ART technique is an iterative approach that works by associating a vigilance parameter with the clusters. The algorithm starts with all points as clusters. Then iteratively cycles through the clusters. The vigilance parameter is a measure of the distance between two clusters required for them to resonate. When two points are close enough to resonate they merge into one cluster for the next iteration. If a point does not resonate with any other cluster then it does not merge. The ART technique is often said to guarantee stable learning. This refers to ART's ability to converge to a solution, which it can always do. One of the drawbacks of ART for the application at hand is that it is sensitive to the ordering of the input vector sequence. One

ordering may produce a different cluster set from another. This is a problem because we would expect that if two independent users request the same set of web pages they would have the same interests, which ART cannot guarantee. There are two different meanings to "stable" learning. In the literature describing ART "stable" refers to its ability to always find a solution. The other meaning of "stable" refers to ART's sensitivity to the data set ordering.

Figure 4.2: ART Sensitivity to Data Set Ordering



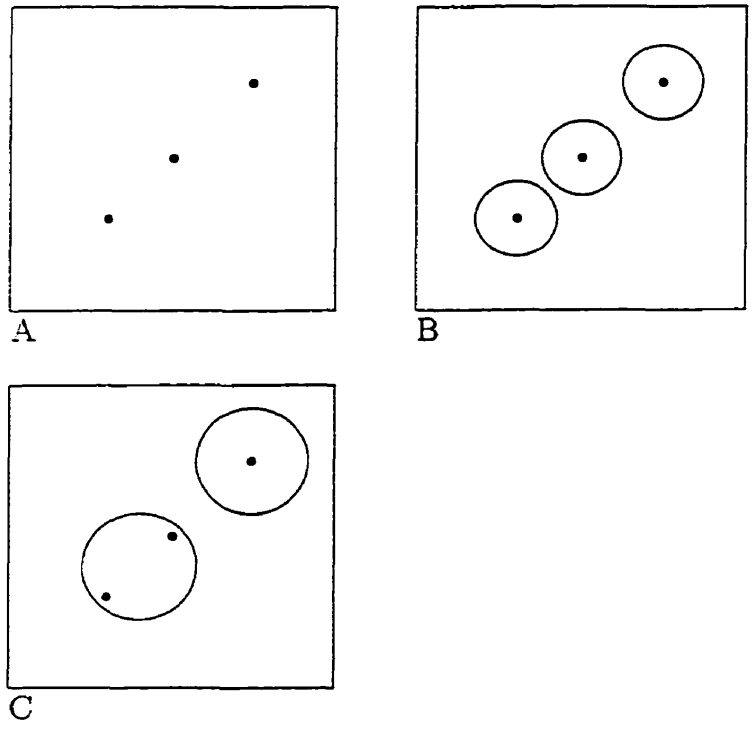
As can be seen from the Figure 4.2, given the 3 points in 2D space Figure 4.2.A shows 3 points that happen to equi-distant. If the points are within the vigilance parameter (Figure 4.2.B) then on the first iteration ART will try to merge the clusters. Unfortunately because they are all within the vigilance parameter the ordering of the data set will decide which of the points will cluster together first. On the following iteration the cluster center becomes the average of all the points in the cluster. Due to this averaging of points the remaining point can no longer be merged. Because of this effect the ART is unstable in the clusters it generates in terms of the input set ordering.

4.5 mART Modified ART

ART's sensitivity to input vector sequence ordering promoted the development of mART by Vlajic [17]. mART uses a dynamically growing vigilance parameter to eliminate the input vector ordering sensitivity. mART starts out by setting the vigilance to zero and making every point in the data set a one element cluster. It then uses standard ART to do the clustering. Once clustering has ended an ART has found a solution. It then slowly increases the value of the vigilance parameter and restarts ART to merge the last re-

sults together. This slowly increasing vigilance enforces an artificial ordering on the data set by forcing ART to only merge one pair of clusters at a time. It also removes the input ordering sensitivity. Unfortunately mART requires the whole data set be available before processing can begin and to add a data point to the set re-clustering must occur. This is required to maintain the artificial ordering in the dataset.

Figure 4.3: mART Insensitivity to Data Set Ordering



By making vigilance parameter grow during the clustering the stability problem can be fixed provided that the vigilance parameter is grown slowly enough to allow only 1 pair of clusters to merge at a time as shown in Figure 4.3.

Unfortunately mART creates another problem for our application. ART has a best case running time of $O(n)$ where n is the size of the input. mART by only allowing the addition of one cluster at a time means that the best case for re-clustering in its naive implementation is now $O(n^2)$. This creates a problem as the data set is typically all of the web pages viewed by a user in the past 6 months often on the order of 1000 pages. As well mART requires a regeneration of the clusters during addition to a cluster set in order to reinforce the clustering stability. To continue the learning process as the user is using the system a regeneration is needed for every page that a user looks at which isn't practical using mART.

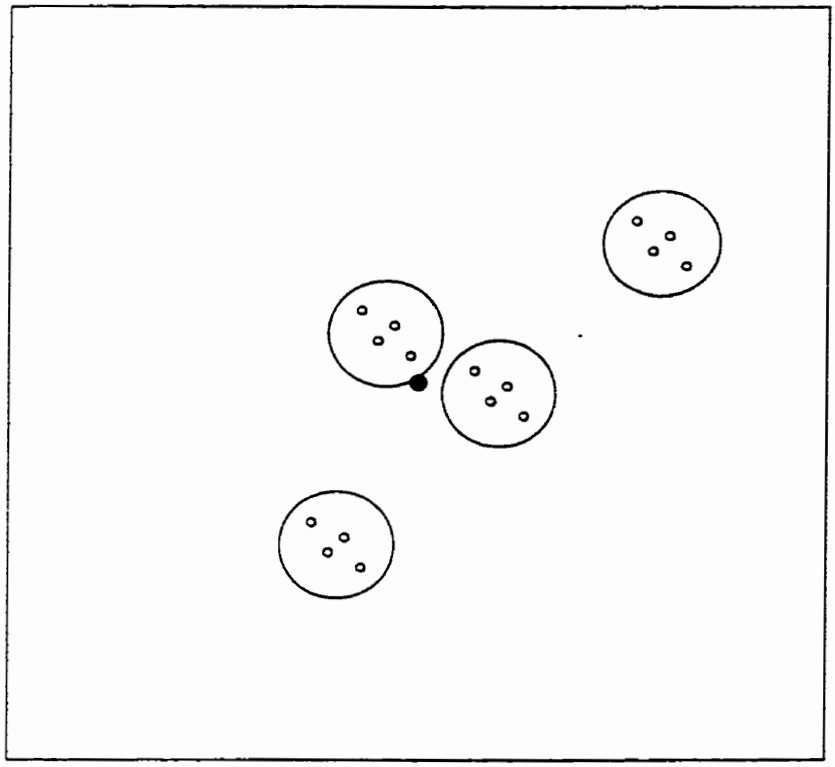
4.6 pmART Progressive mART

mART formed the basis for the development pmART used in our clustering engine. pmART uses the same dynamically growing vigilance parameter but

allows for the addition and removal of data points. This is required to add new web pages and age off pages that have not been revisited. The largest advantage of pmART is that mART requires a full re-clustering in order to add or remove items to the dataset. pmART can add and remove points without a full re-clustering. For our application a full re-clustering using pmART will still take $O(n^2)$. pmART can do insertion and deletions of single points in $O(c^2)$ where c is the number of points in a cluster rather then all of the points in the entire data-structure.

We can achieve this by realizing the addition or removal of a point in a highly clustered space will only affect the cluster that it belongs to. This can be see in Figure 4.4 where the point being added is in the center of the figure. The addition of this point can only affect the two clusters near it. The rest of the space will be unchanged. This means that re-clustering the space is wasted effort. Using this assumption we calculate a region on influence for each point Figure 4.5 . The region of influence for any single point is set of clusters that land within twice the vigilance parameter. During insertion for example, all of the cluster centers that are within the region influence of the point to be inserted are considered. From here only the clusters that

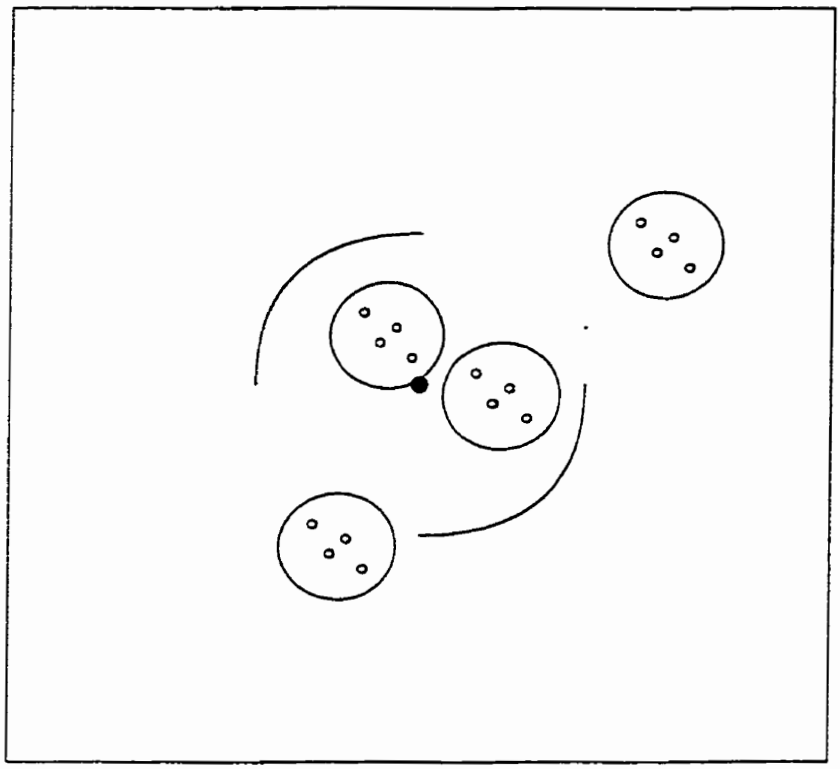
Figure 4.4: pmart Region of Influence



fall within the region are re-grown using an increasing vigilance parameter including the new point to be inserted. Assuming that the topic space has been well defined and the clusters have good separation in the space then the insertion time should only include the re-clustering time of the points in a cluster.

Unfortunately in worst case scenarios where the vigilance parameter is not correctly configured time complexity degrades to $O(n^2)$. This is the

Figure 4.5: pmART Region of Influence



case where the whole space is combined into one cluster and is arguably pathological. In addition for our purposes, the removal of points from the cluster space need not be done in real-time. As such, points can be removed during the addition of new points saving the need to re-cluster for removal.

Chapter 5

Dictionary Generation

5.1 Introduction

In order to achieve good separation between the interest clusters in the data set, the document space needs to be designed to have good resolution in the topic areas that clusters exists. The areas of interest to the user are not known apriori. Techniques for identifying existing topics that provide good separation have been used in previous work [4] [2]. As such the topic space needs to be designed to have good coverage of all the potential areas of interest to a user.

5.2 Static or Start Dictionaries

The broad topic space required for a universal topic dictionary is difficult to form. One proposed technique has been to take advantage of the newer test networks such as Internet2 and CANet. These test networks are being developed as next generation Internets by the Canadian, US and other governments around the world. At this time these test networks are heavily underutilized but have large bandwidth and good connectivity [3] [7]. As such these high performance networks could be used as a low cost subset of the Internet. The test networks could be crawled using a search engine crawler that could use TF/IDF to identify the topic words of high salience for the system. Unfortunately the large majority of the sites on these test networks are universities and government installations. Unfortunately the problem here is that these test networks primarily connect universities and research institutes that may skew the space in favor of an highly academic space. In addition the crawling in general does not really solve the problem of scalability. No matter how fast a network is, crawling may be very time and resource consuming.

An alternative that has been suggested has been inspired by the work of Y. Labrou and T. Finin [19]. They are using the large hand built categorical search engine like Yahoo as an ontology for describing documents. By parsing all of the descriptions or links pointed to by the Yahoo web site we can map the search engines category words to our topic space. This could also be used to enhance the browsabillity of the cluster space by having a mapping of the cluster space to a Yahoo category search tree. One of the draw backs of this is the fact that the space is a hand built ontology. This may lead to a skewing of the space from the builders perception of importance.

5.3 Dictionary Regeneration

A third and probably most practical technique would be to start using one of the above mentioned techniques in stages, dynamically building a dictionary for a next stage while using its current dictionary. The notion here is that the agents would keep track of the statistics and word frequencies of the pages they have seen over some validity period. They would then merge together their statistics at a centralized server. The agents would download a copy of the new dictionary and continue as before. The issue that

arises is that documents that were rendered using one dictionary may need to be rendered in order to extract topics that might have not been discovered previously when using the previous dictionary.

5.4 Universal Dictionary

In a team, web agents will need to able to communicate with each other, this means making additional constraints to the representations used. If agents are going to exchange pre-computed representations of web pages they will need to be using the same topic word dictionary. Fixing a topic dictionary adds a small obstacle as any topic not covered in the dictionary can not be represented by the agents. This stems from the fact that there may exist areas in topic space that an agent may encounter that are not represented in the dictionary. Therefore the agent won't be able to correctly classify the pages. This also makes the case for using a centralized dictionary publishing site to coordinate the dictionary revisions.

5.5 Test Dictionary

For our development, 8 volunteers had all of their web traffic captured for 2 months. The full non-rendered pages were captured to allow development of the required techniques. As such TF/IDF was used across the entire data set to generate the most accurate dictionary possible. This may have skewed results in two ways. First by having a dictionary that was generated across all pages viewed by such a small group may create an over-accurate for the dataset. As well the amount of data used was not representative of the dictionary needed in a true deployment. Unfortunately a much larger data set would be required to improve our results.

Chapter 6

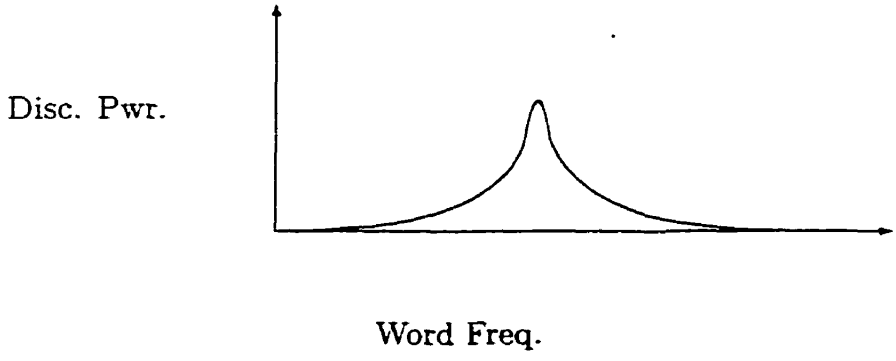
Dictionary Computation

6.1 TF/IDF

In order to identify the remaining root words that are the most effective for the clustering step a standard information retrieval technique is used. It has been indicated that term frequency inverse document frequency (TF/IDF) is effective for this type of work [17]. TF/IDF is used to identify the words that have the highest discriminating power. High discriminating power refers to the word's ability to separate the data set given its existence or nonexistence in the document. In other words a word with high discriminating power would appear in approximately half the documents in the dataset, where as a

word with low discriminating power would only appear in very few documents or almost all documents. Figure 6.1 shows a function of discriminating power versus frequency of appearance in documents.

Figure 6.1: Discriminating Power vs. Word Frequency



6.2 Word Correlation

After finding the words with the highest salience the next step is to remove the words with a high correlation. Word correlation attempts to identify terms in the term dictionary that are separating the space along the same or nearly the same plane. Unfortunately the removal of correlated words was dropped for algorithmic reasons. Discovering which words are correlated involves counting all of the words that appear in the same document as the

word. Pairs of words that appear frequently are said to be highly correlated. Comparing every word in a page with every other word in the page is $O(n^2)$ in both space and time complexity.

6.3 Algorithmic Considerations

In a traditional information retrieval setting the data set is available from the outset of the clustering. This is not the case in our system as visiting the whole data set apriori is exactly what we are trying to avoid. In addition we don't want to overload the client computer with calculations. Our approach needs to take advantage of the distributed nature of Internet users and leverage their numbers. The best way of doing this is to separate the calculation across the group along one of the natural complexity dimensions, web pages. As seen in the previous chapter each client performs a number of tasks while the system is running.

- HTML Tag Removal
- Word Stemming
- Stop Word Removal

- Dictionary Lookup
- Next Dictionary Generation
- Topic Clustering

We need to consider the complexity of each of the steps in terms of the data set that they operate on.

To start HTML tag removal requires only one pass through the HTML file. This is linear in the size of the pages and can be assumed to be in the order of hundreds of words. The stemming operation is constant per word thus linear in terms of the web pages. Stop word removal is no different from a universal dictionary lookup as such it actually rolled into the lookup. A lookup into a trie dictionary is considered to be $O(LogmN)$ where m is the branching factor, in this case 26, and N is the size of the dictionary. As the size of the dictionary is fixed every lookup should take the same amount of time. So in terms of the time it takes to lookup each word in the web page this scales linearly as well with the size of the page as well.

In the prototype implementation, the dictionary is represented as a trie [16] (Assuming ASCII text) data-structure. A trie has a O(LogmN) complexity where m is the branching factor, in this case 26, and N is the number of

entries. N unfortunately needs to be fixed to be the set of all words in the system. Because we are trying to keep track of not only the words in the universal dictionary but also the words that may be considered for the next release of the dictionary.

The result from the pseudo code shown in Figure 6.2 will be a sparse vector of the dictionary words that appeared in the page. This will be stored as a sparse vector because the majority of the words in the dictionary will not appear in most web pages. In addition, the pages URL and an interest level will be included. From this point it becomes a matter of taking the set of sparse vectors, representing web pages, and generating a list of the users interests.

One interesting thing to note is that one of the longest computations that needs to be done is the TF/IDF. As the users are collecting the data in a distributed manner they actually collect and calculate the term frequencies and the document frequencies. Leaving the new dictionary calculations quite simple.

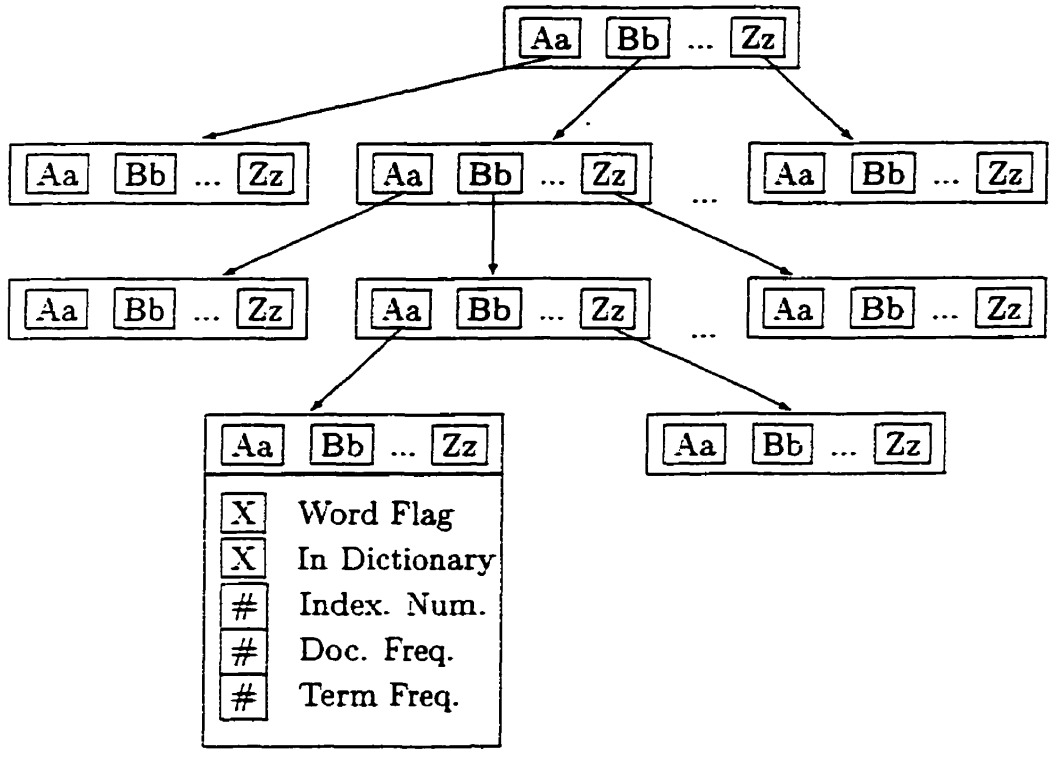
As all of the steps so far scale linearly with the size of a web page they

can easily be done on a client machine as the page is being down-loaded. The only taxing operation is the topic clustering, as pmART scales at $O(c^2)$ where c is the cluster size. The insertion time may vary widely. In our implementation insertions are put into a clustering buffer and a separate thread of operation removes operations from a buffer. This thread performs re-clustering of regions asynchronously from the users web operations. This allows the topic clustering to keep up when insertions are into smaller clusters and to batch operations when the clusters are larger. It also means that if the pmART operations fall behind their efficiency increases as multiple operations to the same cluster can be done with one re-clustering.

Figure 6.2: Web Page Parsing Pseudo Code

```
while file not empty {
      file >> Tag-Removal >> word
      Remove-Stem(word)
      Dictionary-Lookup(word)
      Update word count.  to show its appearance
           // if its the first appearance in the page
           // increment total number of documents counter
           // increment total number of appearance counter
      if (exists in dictionary)
           Add it to the page vector for clustering
      }
}
```

Figure 6.3: Trie Data Structure



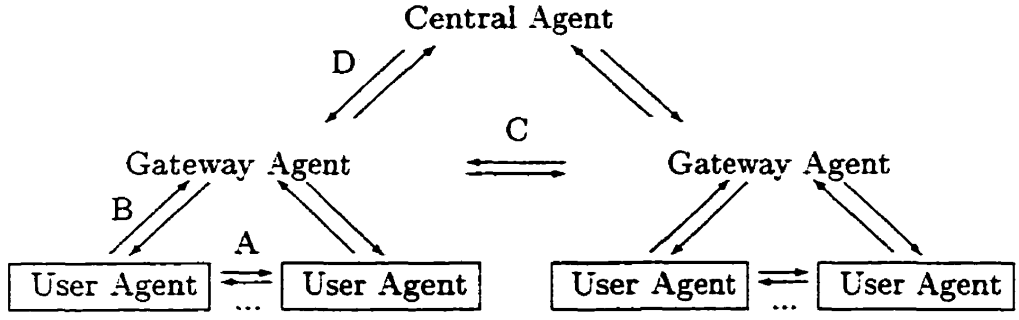
Chapter 7

Multi-Agent Architecture

7.1 Introduction

The design of the multi-agent filtering architecture is based on a distributed network model. The system consists of three types of agents. The agents work together to help to map out the high dimensional topic space of the Internet. Mapping only the areas that are of interest to end users.

Figure 7.1: Agent Interaction



7.2 User Agents

User agents are installed on end user machines. As documents are downloaded the agent parses them and attempts to extract a representation of their content. Over a short period of time the agent builds a database of pages that have been categorized into topics and have an associated user interest level. This database is used by the local agent for learning by the local agent. This database is not leveraged as in traditional web collaborative agents. In a multi-agent cooperative setting, the user agents exchange their database of web pages rather then attempting to re-categorize the pages already viewed by another agent. In this way rather then having many independent filtering agents we have a cooperative team of information agents each an expert in

their user's interest area. In addition pages that have not been revisited are aged out of the database. This keeps the cluster representation of user interests current.

The suggested page results are presented to the user using their browser start screen or real-time refreshing browser window. This leads to a built-in feedback system. As a user becomes interested in a topic he visits pages that discuss it. The user agent notices a new cluster forming and attempts to find other agents with knowledge on the subject. The newly discovered pages are then suggested to the user. If the user visits them they are added to the interest cluster reinforcing the learning. If the user ignores the links the other pages in the cluster age and the cluster fades.

7.3 User to User Agent Communication

User agents are designed to discover partners by broadcasting on the local subnet as shown in Figure 7.1 A. This usually means communicating with approximately 255 other agents locally. This has a lot of advantages, first the agents are geographically close and this tends to make user interests similar, secondly local network communication is usually cheaper. A broadcast con-

sists of a topic vector in the center of an interest cluster. This vector need not represent an actual web page only an interest. User agents may respond to broadcasts if they have web pages that are within the cluster diameter of the request. In this case they simply broadcast the pages they have. If the agent has a interest center within range (twice the vigilance parameter) of the request vector then they enter into a contract with the requesting agent. A contract consists of simply exchanging all points in their respective clusters.

7.4 Gateway Agents

The system also consists of gateway agents. Gateway agents act as local librarians or representatives for user agents to the other subnets on the Internet. They monitor the local subnet for requests. If a user agent broadcasts a request the gateway may respond and enter into communication with a user agent as in Figure 7.1 B. The gateway gives the user agent all of the points it has within the diameter of the request cluster and receives all of the points in the request cluster. The gateway also listens for broadcast points on the network that it adds to its database. The gateway agent does not age the page references it has thereby keeping a long term record of categorized

pages. In this way the gateway has an ever increasing accuracy topic map of the Internet.

Gateway agents are also responsible for building a aggregated statistic dictionary for publishing back to the central registry. The aggregation consists of simple addition and can be piggy backed on gateway agent communication or gateway registry communication.

7.5 Gateway Gateway Communication

The gateway maintains a cache of all the requests broadcast on the local subnet. It periodically contacts other gateways on the Internet looking for references related to the cached requests as shown in Figure 7.1 C. The two gateway agents exchange all cached requests and all data points within the region of the request. This results in the migration of page references around subnets.

7.6 Central Registry

The gateway list server is used as a central list of all the gateway servers on the Internet. Gateway agents contact the list server to update their list of gateway servers as shown in Figure 7.1 D. In order to establish a new gateway, a system administrator need only register with the list server. The registry is also responsible for the generation of new dictionary files. Gateway agents push updated statistic dictionaries to the registry agent during their request for a new gateway list. They may or may not down-load a new dictionary depending on whether or not the registry agent has published one. This is analogous to Napster's list server. Although unlike Napster the system can continue to operate for extended periods without the registry being available.

Dictionary publication is done by the registry agent. The agent collects updated statistics from the gateway agents for the next revision of the dictionary. This happens during periodic communication with the gateway agents. When a new dictionary is published it should be pre-published, that is to say made available with a effective date to ensure all gateway agents are aware of the update.

The generation of new dictionaries is done by recalculating the TF/IDF for the new dictionary. This is based on the new information provided by the user agents via the gateways. As new topic areas emerge on the Internet updated dictionaries will allow the agents to represent them. Areas of the topic space can be expanded to allow for increased popularity. Further study needs to be made into an effective regeneration schedule for the registry agent. The publishing of new dictionaries can be thought of as tuning of the topic space.

Chapter 8

Example Common Clusters

8.1 General Cluster

The dataset consisted of a capture of all HTML documents down-loaded over the course of two months by 8 users. Of the 8 users only 6 down-loaded enough pages to form clusters larger than 3 elements. By adjusting the vigalence parameter two common clusters between 4 of the 6 were found. In the larger cluster the following top 4 pages were found ranked by interest value.

- http://www.yahoo.com/
- http://www.excite.com/
- http://www.altavista.com/
- http://www.dogpile.com/

Figure 8.1: Example Search Engine Cluster Sites

It is interesting to note that the pages in this cluster are probably the most generic pages in the topic space. These are popular start and search pages for users and often contain a very general content of categorical link information.

8.2 Technical Cluster

The second largest common cluster across the group of 4 was of more technical content. This cluster was common to 3 of the 6 people. It should be noted the people that shared in this cluster were all enrolled in the same lecture course on Advanced Internet Working. For the course at the time

a network analyzer was being written as a class project. The pages in this cluster are RFC and FAQ documents describing IP and TCP.

- http://www.cisco.com/univercd/cc/td/doc/cisintwk/itodoc/ip.htm
- http://www.cisco.com/univercd/cc/td/doc/product/iaabu/centri4/user/scf4ap1.htm
- http://www.rfc-editor.org/rfc/std/std5.txt
- http://www.rfc-editor.org/rfc/std/std7.txt

Figure 8.2: Example Technical Cluster Sites

8.3 Sub-Interest Cluster

A third cluster in the dataset was unexpected. A cluster was found between two people that shared a common interest in performance modifications to imported cars. A series of pages linking to various users cars and performance parts manufacturers home pages. The two people involved in this common cluster had no knowledge of each others interest.

- http://www.seko.ca/suz/suzuki.htm
- http://www.calmini.com/swift.htm
- http://www.buschurracing.com/buschur/BRwebsite.nsf/Stage3?OpenPage
- http://www.extrememotorsports.com/g1cat/full.htm

Figure 8.3: Example Unknown Cluster Sites

It should be noted that the vigalence parameter may be artificially large. It was tuned in order to produce 4-6 clusters in the users datasets. With a larger history it may show that a smaller vigalence may be more appropriate.

8.4 Clustering Example Run

As an example Figure 8.4 is the output of the clustering software from the user with the largest dataset. Each point represents one page visited by the user. The braces indicate clusters of pages. It is interesting to note that the

large majority of pages are clusters of single points. These are either pages of very specific content that was only visited once like a news article or a page of no interest to the user. These clusters may grow with a longer time capture.

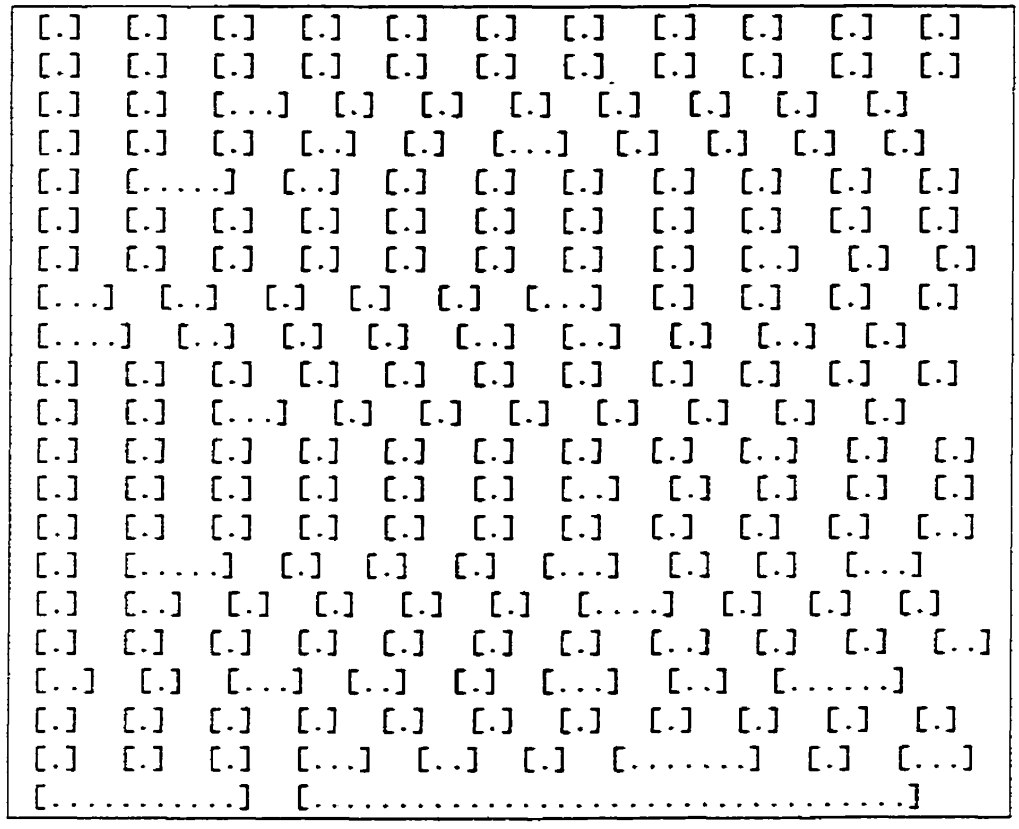


Figure 8.4: Example Cluster Set

Each dot represents a web page viewed by the user, the braces indicate clusters of pages

Chapter 9

Conclusions and Future Work

9.1 Additional System Benefits

Web Traffic Reduction The multi-agent filter has additional benefits than simply aiding users find interesting sites. The system integrates very well with web caching systems as the first sites suggested are from people on the local subnet that should be present in the cache. In addition because users spend less time searching for raw information the total network bandwidth should be reduced.

WWW Search Engine The gateway server also acts as a learning search engine. If a user is searching for a new topic, they can search the database

contained on the gateway agent. This becomes a very effective search engine, because, every day the same search is run, the gateway will have gained additional references, effectively learning a better mapping of the particular region of the Internet topic space.

E-Business Information Having a compact representation of each users interests would be of interest to online e-businesses as it would allow very effective targeted advertising. It would also allow on-line stores to help a user find the products they are looking for more quickly.

9.2 Conclusion

The work described here has been brought to a point where it shows good promise for a potential Internet application. Unfortunately the data set used was taken from a set of 8 users at TRLabs over the course of 2 months. The data set was large enough to allow us to develop the original theories and architectures. To further the work a much larger set would need to be used in order to verify the results so far and to further develop the system. To truly advance the work a large scale deployment would need to be made. This would allow the tuning of high-order parameters like dictionary publishing

schedules. In addition more data would allow for more sophisticated use of the interest parameter.

9.3 Future Work

True web page understanding is a domain that is only in its infancy. Much of the other work being done in the area could be incorporated in to this system to help improve its accuracy. Some of the more valuable data types that would be most useful would be the HTML page structure [14] and image understanding [1].

Bibliography

[1] B. Zimmerman, "A Rhetorical Approach to Understanding Images in the New Visual Age", ACM 1-58113-072-4/99/009, 1999

[2] C. Rijdbergen, *"Information Retrieval"*, Boston Butterworths, 1975

[3] H. Guy, "ARDNOC Traffic Map", http://www.canet3.net/stats/map.html

[4] G. Slaton, C. Buckley, "Term Weighting Approaches in Automatic Text Retrieval", http://cs-tr.cs.cronell.edu/Dienst/UI/1.0/Display/ncstrl.cornell/TR90-1113

[5] S. Lawrence, C. Giles, "Accessibility of Information on The Web", Nature, Vol. 400, pp. 107-109, 1999

[6] U. Shardanand, "Social Information Filtering for Music Recommendation", Phd. Thesis Paper MIT Media Lab, 1994

[7] Abilene NOC, "Abiliene Network Operations Center", http://www.abilene.iu.edu/

[8] AndroMedia, "Movie Critic", http://www.moviecritic.com

[9] B. Sheth, "A Learning Approach to Personalized Information Filtering", Phd. Thesis Paper MIT Media Lab, 1994

[10] Y. Lashkari, "The WebHound Personalized Document Filtering System", http://rg.media.mit.edu/projects/webhound

[11] P. Maes, "MIT Media Lab Software Agent Group", http://agents.www.media.mit.edu/groups/agents

[12] A. Wexelblat, "Automated Collaborative Filtering", http://wex.www.media.mit.edu/people/wex/rate-proposal-ACF.htm

[13] L. Chen, K. Sycara "WebMate: A personal Agent for Browsing and Searching", 1997

[14] N. Vlajic, H.C. Card, "An Adaptive Neural Network Approach to Hypertext Clustering", Proc. International Joint Conference on Neural Networks", Washington DC, JCNN0029, July 1999

[15] C. Bishop, *"Neural Networks for Pattern Recognition"*, Oxford Press ISBN 0-19-853864-2, 1997

[16] D. Kunth, *"The Art of Computer Programming"*, Addison Wesley, ISBN 0-201-03803-X, 1973

[17] N. Vlajic, H.C. Card, "Adaptive Algorithms for Hypertext Clustering", Msc. Thesis Paper University of Manitoba 1998

[18] P. Card, R. D. McLeod, "Improving Data Management on the Web", TRLabs Tech Forum Calgary Alberta, Oct. 1999

[19] Y. Labrou, T. Finin, "Yahoo! as an Ontology - Using Yahoo! Categories to Describe Documents", CIKM 99, Kansas City MO, Dec. 1999

[20] A. Mahanti, C. Wiliamson, D. Eager, "Traffic Analysis of a Web Proxy Caching Hierarchy", IEEE Network, May/June 2000

[21] S. Jones, K. Willet, P. Willet, *"Readings in Information Retrieval"*, San Francisco, Morgan Kaufmann, ISBN 1-55860-454-4

[22] G. Carpenter, S. Grossberg, "A massively parallel architecture for a self-organizing neural pattern recognition machine", Computer Vision, Graphics, and Image Processing. 37, 54-115. 1987

[23] L. Cho, "The Sound of Net Congestion", ABCNEWS.COM http://www.abcnews.com Feb. 27 2000

[24] A. Mancini, "Beyond Copying", ABCNEWS.COM http://www.abcnews.com Oct. 3 2000

[25] K. Hughes, "Swish-E", http://sunsite.berkeley.edu/SWISH-E/

[26] "Hypertext Markup Language - 2.0", ftp://ftp.isi.edu/in-notes/rfc1866.txt, Nov. 1996

[27] R. Osborn, "Knowbuddy's Gnutella FAQ", http://www.rixsoft.com/Knowbuddy/gnutellafaq.hrml

Chapter 10

Appendix

10.1 Appendix A - Stop word list

a about above across after afterwards again against all almost alone along already also although always am among amongst amoungst amount an and another any anyhow anyone anything anyway anywhere are around as at back be became because become becomes becoming been before beforehand behind being below beside besides between beyond bill both bottom but by call can cannot cant co computer con could couldnt cry de describe detail do done down due during each eg eight either eleven else elsewhere empty enough etc even ever every everyone everything everywhere except few fifteen fify fill find

fire first five for former formerly forty found four from front full further get give go had has hasnt have he hence her here hereafter hereby herein hereupon hers herself him himself his how however hundred i ie if in inc indeed interest into is it its itself keep last latter latterly least less ltd made many may me meanwhile might mill mine more moreover most mostly move much must my myself name namely neither never nevertheless next nine no nobody none noone nor not nothing now nowhere of off often on once one only onto or other others otherwise our ours ourselves out over own part per perhaps please put rather re same see seem seemed seeming seems serious several she should show side since sincere six sixty so some somehow someone something sometime sometimes somewhere still such system take ten than that the their them themselves then thence there thereafter thereby therefore therein thereupon these they thick thin third this those though three through throughout thru thus to together too top toward towards twelve twenty two un under until up upon us very via was we well were what whatever when whence whenever where whereafter whereas whereby wherein whereupon wherever whether which while whither who whoever whole whom whose why will with within without would yet you your yours yourself yourselves