

Abstract

The aim for this project has been to visualise the activity on a web site. There are systems doing this, although most of them lack either a dynamic element or a well-structured map of the site. Our implementation consists of two main parts. The static one concerns drawing a map of the web site, which is done in two different ways. The first one takes account of the structural division of the site and the other one, which is more general, is based on a content-based clustering of the pages. The dynamic part focuses on the areas being visited, continuously reading the web server's log file and highlighting on the map the pages most recently downloaded. In the resulting visualisation, there is both a screen version and a version that is meant for public display, making it a part of an intelligent environment. We believe that WebAware in this way can provide useful information to all concerned in the maintenance and development of a web site.

Sammanfattning

Syftet med detta projekt har varit att visualisera aktiviteten på en webbplats. Det finns system sedan tidigare som gör detta, men de flesta saknar antingen ett dynamiskt element eller en välstrukturerad karta över sajten. Vår implementering består av två huvuddelar. Den statiska handlar om att rita en karta över webbplatsen, vilket görs på två sätt. Det första tar hänsyn till den strukturella uppdelningen av sajten medan det andra, som är mer generellt, baseras på en innehållsmässig klustring av sidorna. Den dynamiska delen fokuserar på de områden som besöks – webbserverns loggfil läses kontinuerligt och de senast nedladdade sidorna markeras tydligt på kartan. Den slutliga visualiseringen består dels av en skärmversion, dels av en version som är avsedd att projiceras på en vägg och så bli en del av en intelligent miljö. Vi anser att WebAware på detta sätt kan förmedla användbar information till alla berörda i underhåll och utveckling av en webbplats.

Acknowledgements

There are a number of people we would like to mention as having been of especially valuable help for us in the completion of this project. We start by thanking our supervisor, Lars Erik Holmquist, for having suggested the topic in the first place, and for helping us in the writing process. We would also like to thank the other researchers in the PLAY group at the Viktoria Institute, who have encouraged us and provided technical assistance when so needed. Another 'thank you' goes to two fellow computational linguists, Martin Josefsson and Torgny Rasmark, who carried out several hours of work on one of their programs in order for us to use it in WebAware. Niklas Paulsson, system administrator at the Viktoria Institute, has also been very helpful in providing us with software needed. As native speakers of English, Stephen Hill and Michael Naughton have been great sources of linguistic advice for the writing of this thesis. Finally, we direct our gratitude to our respective family and friends for their loving support.

Contents

1	Introduction	1
2	Background	2
2.1	Related work	2
2.1.1	WebTrends	2
2.1.2	FishFinder	3
2.1.3	Visualizing the Crowds at a Web Site	4
2.2	Foundations of the project	5
2.2.1	Server log measurement	5
2.2.2	Content-based document clustering	6
2.2.3	CyberGeo Maps	7
2.2.4	Information visualisation	8
2.2.5	Calm technology	9
3	Implementation	11
3.1	Static visualisation	11
3.1.1	External structure	11
3.1.2	Semantic structure	12
3.1.3	Drawing the map	18
3.2	Dynamic visualisation	19
3.2.1	Log file analysis	19
3.2.2	Showing visits	20
3.3	Recapitulation	22
4	Results	24
4.1	The map	24
4.2	Clustering	24
4.3	Cluster labels	24
4.4	Division by groups	26
4.5	The system in use	26
4.5.1	Screen version	26
4.5.2	Wall version	27
5	Further work	28
5.1	Tracking visitors	28
5.2	Clustering	28
5.3	Intelligent environment	29
5.4	Textual information	30
6	Conclusion	31

A	Appendix	35
A.1	Web page of Mobile Informatics	35
A.2	Source code	36
A.3	Extracted text	37
A.4	Vector file	37

1 Introduction

Picture the entrance hall or some other public space in your workplace or institution. People pass by when arriving in the morning and leaving in the afternoon, when going to get a cup of coffee, on their way to the toilet, or when just taking a short break stretching their legs.

Now imagine the same room with a work of art hanging on the wall. The passers-by might, at least those new to the picture, stop for a minute and reflect on what they see, enjoy the sight, be horrified at it or stay untouched. If the picture would change frequently, the interest and curiosity would probably increase, as you would wonder what the next picture would be.

Even though this kind of transforming art would be an objective in itself, it would be even more valuable if it provided the viewer with some useful information – ideally something to do with the company or department the viewer belongs to. What we would like is for WebAware to be an example of such 'informative art'.

So, what information would we like to present? As the name of our project suggests, the purpose is to make people aware of the activity on their web site. By 'people' we mean not only web masters and leaders of an organisation, but all personnel in any way involved in what is presented on the site, be it on their own pages or on the ones of their department. By 'activity' we mean basically the sequence of visits to different parts of the site, and the visitors' ways of moving around between them. Our goal has been to display these facts graphically and organise them in a structurally relevant way, making the view at the same time eye-catching and logically intelligible. Awareness of web site activity can, if made use of, bear fruit in maintenance or reconstruction of the site, for example in that you get to know what parts are popular, and what parts might gain from being changed or made visible by links from other pages.

2 Background

Work in the area of web statistics has been done since the beginning of web history, and so it should be no surprise to find several such systems on the market. We have been looking at a few of them and also at a non-commercial project that much resembles ours. We describe them in the section below. In the following section, we will go into the different bases for our project, as well as describe some required background knowledge for further reading of the report. Thereafter, the problem will be specified.

2.1 Related work

2.1.1 WebTrends

The first commercial system we have been looking at is WebTrends Log Analyzer [1]. The analyses it provides are elaborate and detailed and can be done on log files created by any type of web server. The results can be presented in various layout formats and the system also allows for customised reports.

In the examples we have seen of web statistics presented by means of WebTrends, the data are generally shown in three-dimensional bar graphs as well as in charts giving more exact values for each parameter. The information shown is the most requested pages, activity level by hour or weekday, most accessed directories, number of users per number of visits and much more. A typical graph can be seen in figure 1.

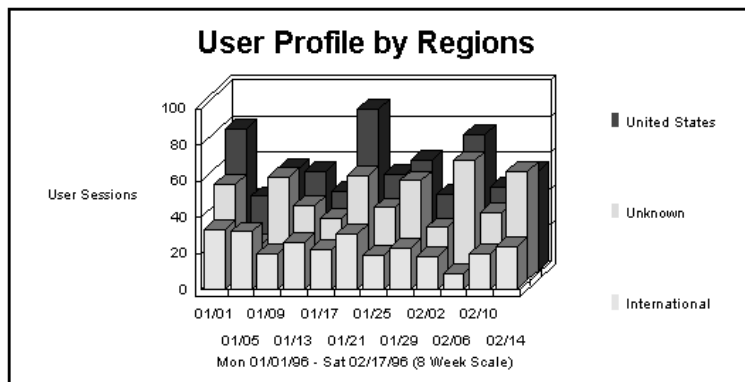


Figure 1: Example of a WebTrends graph

There is an abundance of information available in this form, and together with the flexibility of the system, it makes a product that is clearly useful for many kinds of evaluations of web sites.

The aim for our project, however, is a little different. The above mentioned advantages are obviously desirable, but would be difficult to combine with some of the goals we have set up. For example, if we want the result to be projected on a wall, too much information will cause either crowding on the one picture shown, or frequent shifts between different pictures. The latter would be an option, but could obstruct the purpose of another goal – to constantly update the picture, showing the current activity on the site. With changing pictures, the activity would not be as easy to follow.

2.1.2 FishFinder

The fishFinder is an applet that can be found on the Java Developer Connection site [2]. The purpose of the applet is to visualise crowds of people visiting the different areas of the web site and thus be a help when navigating it. The crowds of people will draw your attention to them, and will make you want to go there to check out why all the people are there.



Figure 2: The fishFinder applet

It is limited in the way that it shows only the four most visited areas of the site at a time, and to update the display you have to push a button.

The applet uses a radar/fish-finder metaphor to show you where the action is, a feature that we too would like to capture in our system, only we would like to combine it with a more detailed site map. Another important thing we would like to capture, that the fishFinder applet misses out on, is dynamics. In its present form it is really nothing more than an excerpt from a bar graph, crammed into a metaphor.

2.1.3 Visualizing the Crowds at a Web Site

Nelson Minar [3] has created a visualisation of the crowds visiting a web site. His visualisation is an attempt to capture the crowd dynamics of the visitors, so that the viewer is able to identify the popular parts of a site, see the paths connecting them and (as Minar puts it) 'get a feel for the overall activity on the site'.

Minar distinguishes three problems that have to be solved to create an effective visualisation:

Map In creating a map of a web site, Minar has chosen to let a single icon represent a group of pages and let the colour of the icon represent to what kind of group the icon belongs. The placement of the pages is hand-constructed and Minar admits that an algorithmically constructed map is preferable.

Individuals You have to find a way of representing individual people in the visualisation, although each individual plays only a small role in a crowd. Minar has chosen to simply represent each individual with a small dot, coloured with respect to its domain-name.

Crowd dynamics To visualise the crowd dynamics you have to create an animated display. In Minar's visualisation you can see people moving around the site in accelerated time. The acceleration serves to enhance the patterns in the activity.

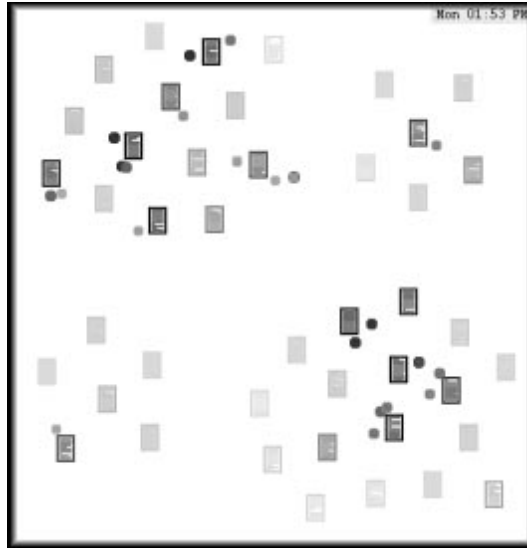


Figure 3: Minar’s visualisation

To further emphasise the active parts of the site, persons and pages are gradually faded when not involved in any action for awhile, which makes the more frequently visited parts stand out from the rest. This is an interesting feature that we would like to incorporate in our system, although we do not wish to attach as much weight to the visitors as to the pages and the information they contain. We also would like to incorporate the ability to make all of the pages visible at the same time rather than letting a single icon represent several pages.

2.2 Foundations of the project

2.2.1 Server log measurement

The main information source when it comes to web statistics is the record of downloaded files which the web server keeps in one or more log files. These come in different formats, but in general the most basic information logged is the date and time when a file is downloaded, the IP number of the requesting computer and the name of the file.

The site we have been working on is the one of the Viktoria Institute, where our project was performed. Its server log is in the Apache Common Log Format, where each line consists of these data:

```
host ident authuser date request status bytes
```

The meanings of the line entries are the following:

<code>host</code>	domain name of client, if available – otherwise IP number
<code>ident</code>	identity reported by client (provided an identity check is enabled)
<code>authuser</code>	userid for password protected document
<code>date</code>	date and time of the request
<code>request</code>	request line from client
<code>status</code>	three digit status code returned to the client
<code>byte</code>	number of bytes in the object returned to the client

When a value cannot be found, the entry is represented by a hyphen.

So, what is it that we can extract from these data, and what do we use it for? In a dynamic application, like we intended WebAware to be, the time parameter is of special importance. The fact that we get the exact time (e.g. 06/May/1999:10:22:52 +0100) for every file request is therefore of great value. It enables us to process not only at what time a page is or was being visited, but also how long it was since last time, how many times it has been visited since a certain time back, and so on. One thing that cannot be found out from the log file, however, is at what time users leave the pages they have visited. Information of that kind would be of interest, as web site owners would then get to know how long time users spend on the individual pages.

2.2.2 Content-based document clustering

In the report for DropJaw [4], a project about classifying web documents with respect to genre and topic, clustering is described as a way of finding 'natural groupings among a set of elements'. In other words, it is a kind of classification or categorisation.

You only need to go to a library, or your own bookshelf for that matter, to realise that organising and grouping of texts can be done in many different ways. They can be categorised with regard to author, title, genre, content, size or indeed any attribute whereby a text may be said to either resemble or differ from another.

In content-based categorisation, the primary question is: What constitutes the contents of a text? This question will then be followed by others, like: How can the contents be measured? and How will a comparison be made possible?

A successful method, used in information retrieval, is one based on term frequency (again, see the DropJaw report [4]). It is based on the statement that terms with neither too high nor too low frequency appropriately represent the contents of a document. For content measurement, instances of word types are counted and a 'term vector' is created, where each term in the document keeps one position. Highly frequent terms considered as giving no contribution to the topic (such terms as 'it', 'but' and 'and'), are ignored, usually by means of a list of these very common words. There are also ways of taking account of the length of the documents to be compared, and of terms that are frequently used in all of the documents, hence not distinguishing one from another. After these considerations have been regarded, the final result is a vector for each document, containing values for the word counts. The set of texts are then compared and grouped according to the relative similarity of their corresponding vectors.

2.2.3 CyberGeo Maps

The work from which the WebAware project emerged was performed at the Viktoria Institute in 1998 by Holmquist, Fagrell & Busso [5]. It is called CyberGeo Maps and aims at graphically presenting the state of a web site regarding the age and size of the pages as well as their position in the directory hierarchy.

This was done by comparing a web site to a solar system, the root directory being in the middle with items revolving around it in different orbits according to their respective depth in the directory structure (see fig 4). In this way, each page on the site is represented by a planet in the galaxy and the size of the planet on the drawn map pictures the size of the file in question. The age of the file is shown by the relative shading - the more recently modified the lighter.

We found this to be an appropriate metaphor for picturing the geography of a web site, since it gives the hierarchy of the directories in an intelligible way, but also because it gives an impression of the massiveness and mystery that you could say are significant for both space and cyberspace.

The one thing about the map that we want to do differently is the manner in which the representations of the pages are distributed on each directory level. In the CyberGeo Maps project, it is done by means of a hash function based on the name of the document and the directory it belongs to.

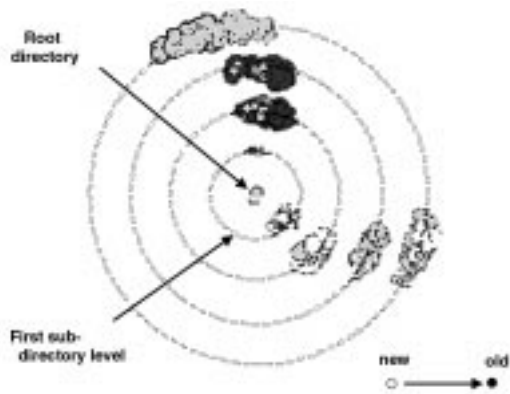


Figure 4: A CyberGeo Map

Documents in the same sub-directory would by this be placed in approximately the same area of the circle. Regarding the semantic aspect of the distribution of pages, we would like to try another kind of categorisation, based on the contents of the pages. We would also like to make it more obvious to the viewer on what principles the grouping is based.

2.2.4 Information visualisation

Great parts of the information that interests us in our daily lives are well hidden in a thicket of abstract data. We need a machete to cut ourselves a path to access the temple of information.

For smaller data sets, tables and charts are fairly good ways of making the data easier to interpret, but as the set of data grows bigger, it becomes harder to grasp the whole picture and thus a need for some kind of a survey of the set is created.

Information visualisation is a way of providing this kind of spatial survey of the data set, often combined with the ability to zoom in on the parts you are interested in. Finding the right metaphor is crucial to information visualisation, since a familiar context can make relationships and structure within data become apparent to you and can also help you understand these data. (For readings on this topic, see Card et al. [6].)

There are numerous subfields to this area of research, of which we have especially devoted our attention to one, namely the following:

Web-based information visualisation

The massive quantities of information accessible on the Web, and the need for means to survey this information, have made web-based information visualisation one of the most prominent subfields of information visualisation.

Rohrer & Swing [7] separate three structures inherent in web-based information:

web topology which denotes the hyperlinks between pages and the intra-document structure,

external structure which is a logical grouping of the pages, for example the directory hierarchy on the server, and

cognitive models where you try to group pages according to how they are related, for example content-based clustering.

By using abstractions of these inherent structures when creating a visualisation, you can significantly augment it and thus make it easier to comprehend.

2.2.5 Calm technology

In an era of information drowning us in demands for our complete attention, we need to somehow single out what is to be attended to, so as not to collapse in stress and confusion. Weiser & Brown [8] suggested a means for this by introducing a new approach to technology design which as well as being informative would at the same time be encalming.

The basic idea is to place things in the periphery that will provide us with continuous information, received more or less subconsciously. We are attuned to it but it does not take our full attention. However, it may do so when needed, making itself known in one way or another. In this manner, focus is easily shifted, and what things are central change according to priority.

The way in which WebAware can be regarded as an application of this technology design is that it is meant to be projected on a wall, practically as part of the fittings. From that position, it would constantly present information about the state of activity on the web site in question and give notice as soon as there is a change of state worth observing (like for example a new visitor arriving). In other words, it would be an information source placed in the periphery with the possibility to gain and regain its position in the centre of attention, and thus be a part of an encalming technological environment.

Specified problem

Now, what we would like to achieve with WebAware is a web site analyser that draws a map showing the current activity not as much in exact numbers as in a graspable way. The problem can be split into three parts:

1. The system should be as general as possible, managing to analyse sites small and large alike.
2. Possible relations between items should be obvious to the viewer simply by looking at the map.
3. The visualisation should be dynamic, showing at every moment what is going on right then, as well as giving a limited trace of history.

There is more than one way of structuring data, and different structures meet different needs.

We believe that a content-based structuring of the map will work for the benefit of the second issue. The text on every page should be analysed, and representations of pages dealing with related topics should be on shorter distance from each other than pages that are different in contents.

Apart from this map design, we want to provide at least one more, based not on content but on external structure (see Web-based information visualisation under Section 2.2.4). There are several such structures to choose from, and thus we could produce multiple map views, but as the scope of this project goes beyond just drawing maps, we have decided to limit the number of map views to two. In choosing the external structure of the map design, we intend to take into account what information users are mainly interested in and make use of any natural divisions on the site.

3 Implementation

The two main parts of the implementation are the static visualisation and the dynamic one. The static gives a picture of the whole site, only changing when new pages are added or old ones edited or removed, while the dynamic focuses on the parts being visited, changing shape continuously. We will describe our methods for implementing the two in separate sections below.

The software tools we have used have been chosen for the specific tasks to be performed. For text handling, we found Perl the most useful programming language, and correspondingly Java for graphics and for keeping it all together. The tools for semantic structuring will be introduced in Section 3.1.2.

3.1 Static visualisation

First of all, to be able to draw a map of the site, we need to know what pages are on it. For this purpose, we wrote a Perl script that searches recursively through the web directory, returning a file listing the full URL:s. This file is then used when looking up the contents on each page (see Section 3.1.2).

The two map designs we have implemented are based on different principles regarding the distribution of the pages. The first two subsections describe these, and the third addresses the map drawing procedure.

3.1.1 External structure

The first map view is based on the external structure of the site (see Section 2.2.4). When creating the map we studied the site, trying to find the most natural way of dividing it into different areas. We did this both by browsing the site with a web browser and by examining the directory structure on the web server. We found that the best way to divide the map was the division more or less already made by the site creators, that is, dividing it into the four different research groups and an additional group with miscellaneous pages containing general information about the institute, publications, personnel etc. (See fig. 5.)

Unfortunately, the pages belonging to each group vary greatly in number, making parts of the map quite crowded while others are significantly more sparsely populated. We still believe this to be a more natural way of dividing the site than some other, possibly more even, distribution of the pages.



Figure 5: Division of the site map

3.1.2 Semantic structure

In our second type of map structure we have used semantic criteria to group the pages on the site. More specifically, we have used content-based clustering, with the help of WordNet [9] – a lexical reference system for English – and a clustering algorithm implemented by Josefsson & Rasmark [10]. We describe the procedure in the sections below. For an example of results from the first steps, we refer to the appendix.

Fetching contents

For defining the contents of a web page, we have to consider primarily what the viewer actually sees when downloading the page. The words of the text usually gives the most information, but there might also be pictures and links, for example, contributing to the contents. How to deal with these and other non-linguistic parameters is a slightly more complicated question than the text issue, but we do take it into account, if only to a limited degree.

Firstly, we have to have access to the source code of each page, which is given by means of the `get` command in the `LWP::Simple` package in Perl. After having fetched the code, we use Perl to clear it from HTML-tags and other extras, so as to get only the text contained on the page. To prepare it for WordNet (see specific section), this Perl script also clears the text from words with strange characters.

To emphasise the main information on the page, possible keywords are picked from any meta-tags found, and so are all alternative texts for pictures.

(For more information on HTML, see the W3C web page on this topic [11].) To get an overall measure of what the page contains, we also count the number of images, e-mail addresses and links that are on it. The links are divided into intra document (within the page), intra site (to other pages on the same site) and inter site links (to other sites), and there is a count performed for each kind separately.

There are two exceptions, where this procedure is not executed. The first one is when the page in question cannot be fetched. The second is when the file contains frameset tags. This latter exception is based on a claim of our own, that these pages contain no additional information to the one on the pages they hold together.

In the ordinary case, a text file is produced for each of the pages fetched, containing the number of images, links etc. along with the whole text including keywords. The URL for the page is saved in a file, just like when they all are first collected (see introduction to Section 3.1). However, in the case of exceptions mentioned before, the URL in question is put aside and excluded in the new, up-to-date list of the site's URL:s.

Creating vectors

To compare two or more documents semantically, there must be comparable representations of the contents. As mentioned in the previous section, the contents we have to deal with are, besides the results from counting images, e-mail addresses and links, the words of the document.

The numerical values from the counts can easily be compared to values for other documents by putting the values in a certain order in a list for each one of the documents, and then comparing the lists.

What about the words, though? What kind of mill will they have to pass through to be of a form that is as easy to compare? In the background chapter we mentioned a method for this task – the term frequency method, counting instances of word types and placing the result in a vector, in the position of the specific type. This would most likely be an appropriate way of numerically representing the lexical contents. However, because of the abundance of existing words, the vectors would be large, possibly unmanageably so, and how could you ever take account of all word types that are or could be in use?

The method we have chosen is not as problematic in that sense. Instead of counting instances of word types, we let every word be assigned to one or more of a fixed set of superordinate concept classes. Each of these classes, then, takes up one space in the vector.

WordNet lookup

For the task of finding concept classes and assigning them to words, we used WordNet [9]. It is an electronic lexical database for English, where words are organised in terms of their meanings. Synonyms, antonyms and hypernyms are examples of information available on the words. In this respect it is structured like a thesaurus, linking concepts to other concepts based on their semantic relation.

After the pure text has been taken out of a page as described earlier, all words are one by one being looked up in WordNet with regard to hypernyms, i.e. superordinate, more generic concepts. We chose 250 such concept categories in our classification of the documents. To appoint these categories, we used a rather simple method:

Out of the collection of pages on the site, we selected ten of the longest texts, looking up every word in each of them in WordNet and saving the results to a file. We counted the occurrences of the different concept categories and appointed 250 of the most commonly occurring.

In our WordNet-lookup script we then created a hashtable with the categories as keys. When looking up the words in the texts extracted from the web pages, each occurrence of a word in a specific category causes the value associated with that category key to be increased by one.

What we get in the end is a number of 255 positions long numerical vectors, each one describing the contents of a document. The first five positions in the vectors contain the results from counting images, e-mail addresses and links, and the remaining 250 are WordNet category entries.

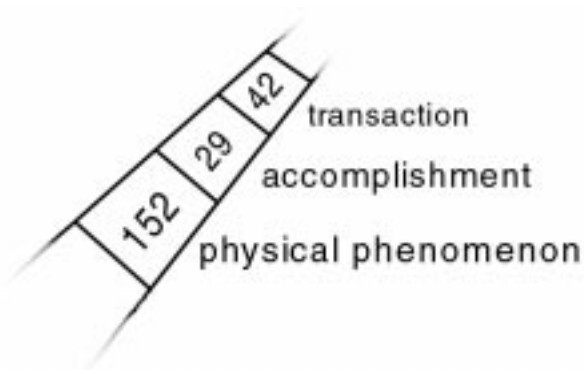


Figure 6: Fragment of a possible vector

Clustering

When having created a vector for each document in the collection, the comparison and grouping are made by means of a clustering algorithm. It was authorised for us to use by Josefsson & Rasmak, who implemented it within the frames of their Master's thesis.

The principle they have used in comparing one vector to another is to measure the angle between them. Initially, every vector represents one cluster. Then the pair of clusters with the smallest difference in angle forms a new cluster and so on in a recursive manner (see fig. 7), until the remaining number of clusters equals or falls below the desired number. We will not go closer into the details here, so for more information on the algorithm and research in the field, the interested reader is directed to Josefsson & Rasmak [10].

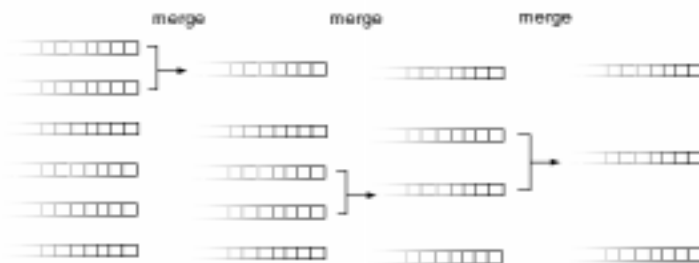


Figure 7: The clustering process

The result we get from running the algorithm on the pages of the site is a predetermined number of clusters, where the documents in one cluster are more similar in contents to each other than to documents of other clusters.

The optimal clustering would be one where, for each page, a comparison is made to every other page on the site. The result from such a clustering would most accurately show the internal relations between all the pages. However, the procedure would be extremely expensive in time and would not be possible for that reason.

Some kind of partition had to be made for the algorithm to be able to cope with the amount of pages on the site. Our first intention was to divide the clustering between the different directory levels, so that the pages at one depth in the structure would be grouped independently of the grouping at other depths. This did not prove to be an efficient way to go about the problem, since the time consumed for the clustering and labeling (see

next section) did not increase linearly with the size of the directories. This naturally caused trouble with those levels containing large amounts of pages. We had to find another way.

For every new cluster being formed, a corresponding vector is created to represent it. This makes the nature of the clustering recursive, in that it can be performed in stages, not having to be done all in one go. This gives us the possibility of dividing the procedure into smaller steps – a precaution recommended for problems of high complexity. Another advantage this brings is that, because of the gradual clustering, the initial division by levels (or by any manually constructed group of pages) does not have to be made. We can take account of the semantic connections regardless of levels, and a classification closer to covering the whole site can be achieved.

There are still restrictions, however. A complete comparison between all the pages on the site is not possible due to the complexity of the algorithm, and so we need to separate the pages into groups within which a comparison can be made. The clusters in all of these groups are then indissolvable; pages can be added, but not removed. We seem to end up with a similar restriction as the one of the directory levels – although this method crosses visible borders and the first one does not. Also, there is one important difference regarding complexity. With this latter method, we are able to control the amount of vectors to be compared in one group, always making sure the amount is manageable.



Figure 8: Displayed result of a clustering

Labeling clusters

The clustering would not be of much interest unless there was a way of telling what it is that signifies the documents of a particular cluster. This is why we have decided to label the clusters and attach the labels to them on the map. We already have a representation of the semantic contents of a cluster, since a new vector is created whenever a cluster is formed. The only thing is to find what separates one vector from those of other clusters.

We have done this by looking at the semantic categories that are most frequently used in each cluster, and comparing them to the most frequently used in the others. The categories in question are the ones that we have selected from WordNet and from which the vectors are built. It is also these categories that will play the parts of labels.

If the category with the highest value in the vector, i.e. the most common one for that cluster, does not have the highest value in any of the other clusters, then this category is instantly given as the label for the cluster in question. If, on the other hand, the most common category for one cluster coincides with the most common for any of the others, there is a comparison made among the second most common ones, and so on, until we have a unique category for each cluster. We have illustrated this procedure in fig. 9.

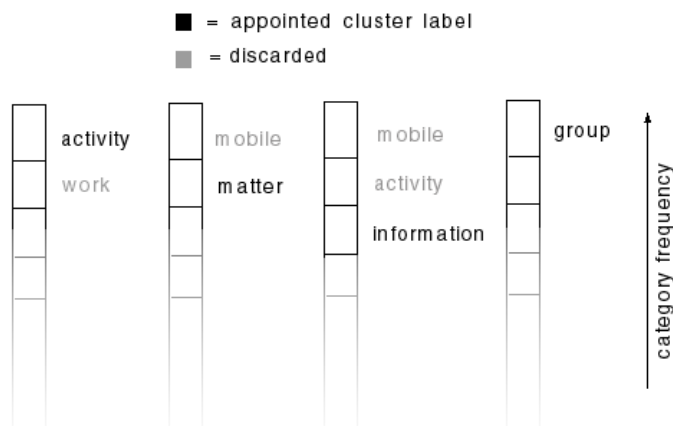


Figure 9: Finding unique labels

To prevent the labels from being too generic, not giving satisfying information about what the clusters contain (for example a cluster having the label 'physical object' or 'abstraction', both of which could describe the contents of many a cluster), we take away the ten most frequent categories from every

vector before we start comparing. This might seem like a somewhat rough and crude method, and that may be true, but unfortunately time has not allowed for exploration of different modes of procedure.

3.1.3 Drawing the map

The idea behind the maps is to use the same kind of solar system metaphor as used in the CyberGeo Maps project (see Section 2.2.3), where each 'celestial body' drawn on the map corresponds to one page on the site. Its distance from the centre indicates at what depth in the directory structure the page lies – hence each directory level will correspond to one 'orbit'.

The algorithm

The site is first divided into groups, in either of the ways described in the two sections below. Each of the groups is then allotted a sector of the circle within which to place its pages. So, for example, if the site is divided into six groups, each of them will have $360/6 = 60$ degrees of the circle.

External structure

When drawing the map based on the external structure of the site, we started by dividing the pages into five parts – the four research groups and an additional one consisting of miscellaneous pages not belonging to any of the other groups. Since two of the groups consisted of a significantly larger amount of pages than the others, we allotted to them 90 degrees each, leaving 180 degrees for the other three to share (see fig. 10).

For each group, we spread the pages on every directory level in a similar way to how the whole circle was divided earlier. E.g. if a group that has been allotted 60 degrees of the circle has six pages on one directory level, it will split up the 60 degrees so that each of the pages gets a space of 10 degrees.

To discern a page in one group from pages in other groups, they all have a specific colour according to the group they belong to. This is especially important in the semantic map structure, where the group divisions are dissolved.

Semantic structure

The task of finding coordinates for the semantically structured map is similar to the one for external structure. We based our solution on the same principle as described above, the difference being that the groups we are dealing with in this case are semantic clusters and not handmade divisions. At the present

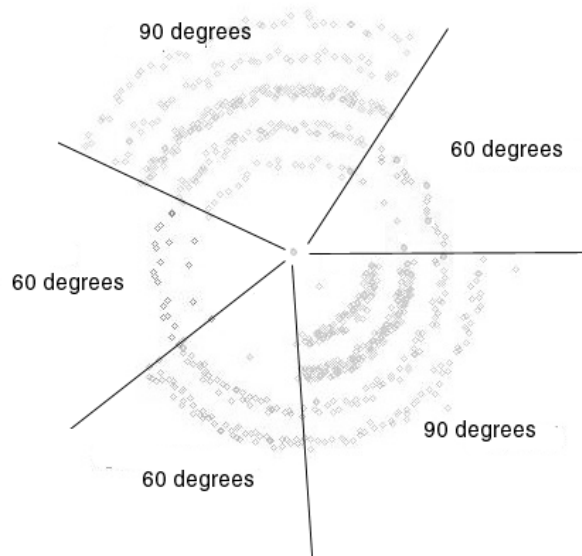


Figure 10: Division of the Viktoria site map

time, however, some work is done by hand, since one of the clusters turned out to be significantly larger than the others. In the same manner as above, we decided to allot this cluster a bigger piece of the cake. This problem could be solved by algorithmic means, but time consuming work on other parts of the system forced us to use this kind of ad hoc solution.

3.2 Dynamic visualisation

The dynamic element of the system is the continuous redrawing of the map as time passes and pages are requested. To keep track of the requests made, a constant check of the web server's log file is needed. We describe this and the way in which we show the results in the sections below.

3.2.1 Log file analysis

The Apache Common Log Format described in the background chapter can be configured to fit various needs that web site holders might have. The Viktoria Institute, owner of the site in this case, has chosen to modify it only slightly. Of the previously mentioned data (see Section 2.2.1), only the host is different, in that it is always represented by the IP number of the client.

An example of a line in the log file would be

```
130.241.141.244 - - [03/Feb/1999:10:23:07 +0100] "GET /groups/play/ HTTP/1.0" 200 185
```

The file formats logged are naturally all kinds available on the Web – first and foremost .htm and .html but also .pdf, .gif, .jpg etc. The only files we are interested in, however, are the actual pages, i.e. the .htm and .html files. Therefore we want to pick the lines in the log file which contain names of files of those formats, and also the ones ending with /, as in the example above, referring to index.html. Out of those lines, we then want to extract certain data on each request.

The picking of the lines constituted a small problem, since the log file is located on the server, running Solaris, while the main Java program runs on a Windows 95/98 workstation. Fortunately, however, the relative platform-independence of Java allowed us to implement a client/server-based solution to the problem. The thought behind this approach is that the server program does the computations and then sends the results to the client programs connected to it.

The server part, running on the UNIX machine, uses a c-shell script to read the 200 last lines of the log file and single out the ones of interest to us (i.e. the ones containing .htm and .html files). These lines are then processed further, so that the only information being sent to the client is the URL and the time when it was fetched. This procedure is repeated every five seconds, which for the amount of traffic on this web-server is more than enough.

The client part, running on the workstation, simply reads the URL and time and checks if any of the visits sent to it are new since the last round. If there are such data, they are sent to the visualisation part, and eventually, the new visit is shown on the screen.

3.2.2 Showing visits

For each visit paid to the site, we wanted it to be made known at the moment it occurs. To draw attention to the page being requested, we have chosen to highlight it in a clearly visible colour, distinct from the background colour and all other colours on the map. Then, as a picture of the amount of time passed since the visit, the colour of the page changes, slowly assimilating to the background, until the page eventually regains its original colour – the colour of its group.

When the client detects a new request in the data received from the server, it immediately notifies the page (or rather the representation of the page), and the variable storing the latest download of the page is updated.

The next time the display is repainted (which is done with a frequency of 20 times per second), the page will be shown in the brightest colour.

An example of what the map might look like can be seen in figure 11 (arrows and explanations added).

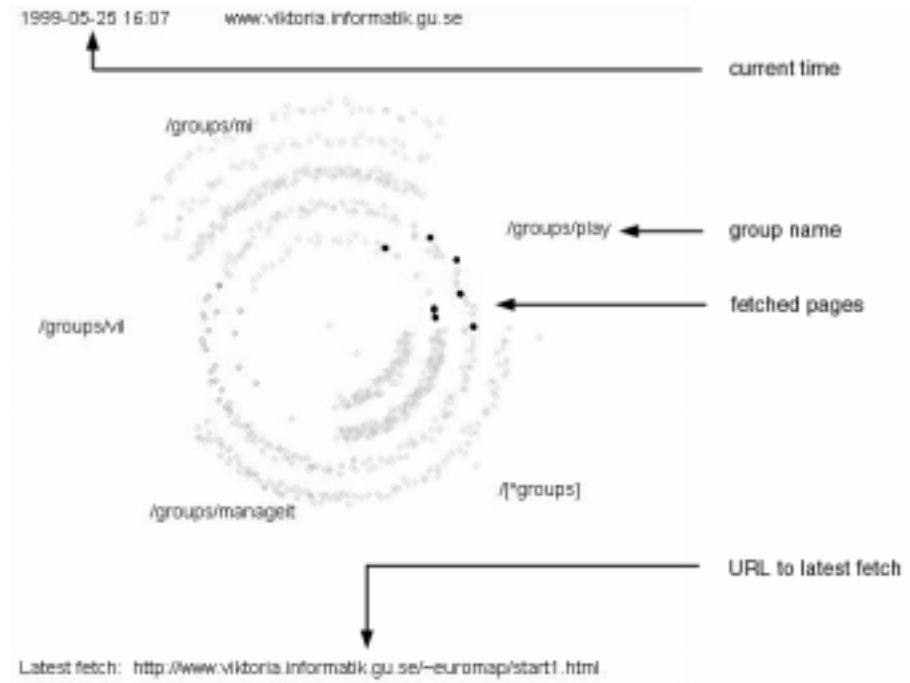


Figure 11: Screenshot

3.3 Recapitulation

To summarise what has been said in this chapter so far, we will now present the implementation procedure in the form of a schematic figure, showing the major parts of the implementation, accompanied by explanatory comments. (We refer to the appendix for an example of a web page, its source code and the corresponding extracted text and vector.)

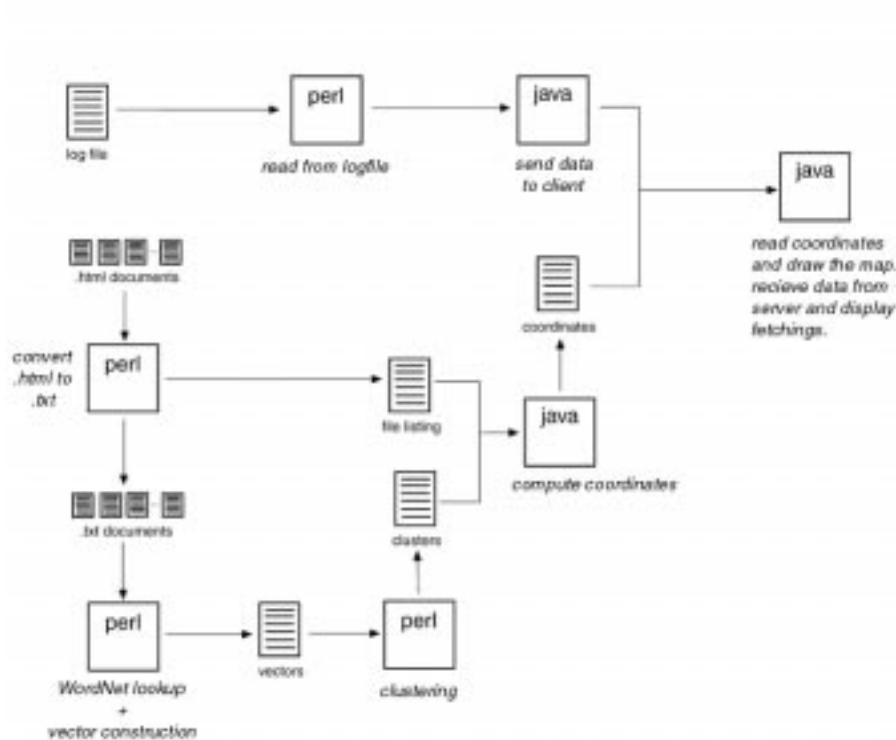


Figure 12: Schematic picture of the implementation

One part – the one we have referred to as the static one – starts with a number of .html documents, namely all the non-empty ones on the site. We process these files with Perl, returning the files with all HTML-tags removed, and with the new suffix .txt. Another result of the process is a list of all the HTML-files, to be used at a later stage. The text files are processed further, by means of Perl and Wordnet, to result in vectors representing the contents of each document. The vectors are then compared, forming clusters which, together with their respective labels, are written to a file. This file, together with the listing of the HTML-files mentioned earlier, form the input to a

Java program, which calculates coordinates for where on the map to place the page representations.

The part we have referred to as the dynamic one begins by a Java program starting a Perl script that reads the log file. The script then writes its output back to the Java program, which immediately sends it on to another Java program running on a workstation. This latter application draws the map on the screen and our work is done.

4 Results

We will now point to some of the results of our implementation that we believe are worth noticing, and then we will give a short description of the two main versions of the system.

4.1 The map

In the visualisation of the site, we found the solar system metaphor appropriate for picturing a delimited area of cyberspace. Compared to the ordinary picture of a tree structure (which in fact is what we are dealing with here), a circular structure gives more room as the levels increase in size, and it does not as obviously include the false assumption that the units would be organised also in the horizontal plane (speaking in terms of tree structures).

4.2 Clustering

The feature of our system from which we had expected most was unfortunately also the one that failed us most. Our first attempt at a depth dependent clustering gave a satisfactory result for most of the levels, with clusters similar in size distributed evenly, giving a symmetric and organised impression (as can be seen in fig. 8 under Clustering in Section 3.1.2).

However, because of the high complexity of the algorithm, we had to leave this method behind. For the levels with the largest amounts of pages, the script did not terminate but crashed due to lack of storage.

The currently used method, based on dividing the site regardless of levels and performing the clustering recursively, meets no problems whatsoever in the storage issue. The problem is the result we get. Of a total of around 850 pages on the site, more than 700 are placed in one huge cluster (see fig. 13). This is clearly suspect and should be the subject for closer investigation.

4.3 Cluster labels

The results from labeling the clusters are difficult to evaluate, since the clustering itself has probably not been a very accurate one. The fact that one cluster contains the majority of pages is likely to have effect on the label assigned to that cluster, and also on the labels of the others, since they are dependent on each other.

If we still want to say something about the results, we could say that they were satisfactory in that they provide the viewer with information that

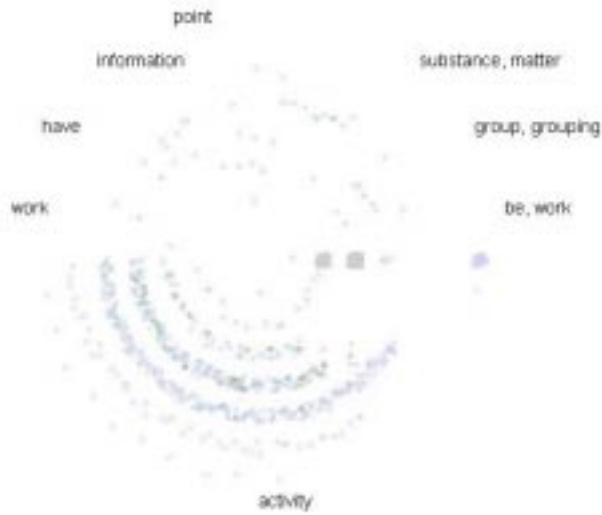


Figure 13: Result of level independent clustering

otherwise would not be visible. With no labels, nothing could be said about the contents in the clusters, and the grouping would make no sense.

We could also mention the advantage of creating labels from categories already available for another purpose, namely the purpose of constructing vectors for the clustering. This is a way of recycling data and thereby enhancing processing efficiency.

Looking to the disadvantages, the labels are more generic than we would have liked them to be. As shown in figure 13, a cluster can have such a label as 'substance, matter', 'work' or 'activity', which is not what we would call informative. It is not certain whether this imperfection is due to badly chosen categories, a faulty method in finding labels, or some other reason. We would argue that the idea about using superordinate concept classes from WordNet is a good solution, but it should be investigated as to how the method could be improved.

4.4 Division by groups

We find that the map view based on external structure works quite well. If you are familiar with the web site of the Viktoria Institute, you easily recognise the four reasearch groups, and can follow the activity on these pages.

The one thing that constitutes a problem here is the labeling of the fifth group, containing the pages that does not belong to any of the research groups. It is possible that one could find a way to divide this group into subgroups, thus making it easier to label. However, we have been unable to find such a division.

4.5 The system in use

The final product of the implementation comes in two versions. One is a screen-based version and the other one is designed particularly for public projection.

4.5.1 Screen version

In the screen-based version, the display window is divided in two, one half showing the map of the site and one having buttons for choosing between semantic and structural focus. Not all of this second half of the window is used up, why we consider the rest of it being a suitable place for providing additional textual information about the site activity. This, however, goes under the section of 'Further work'.

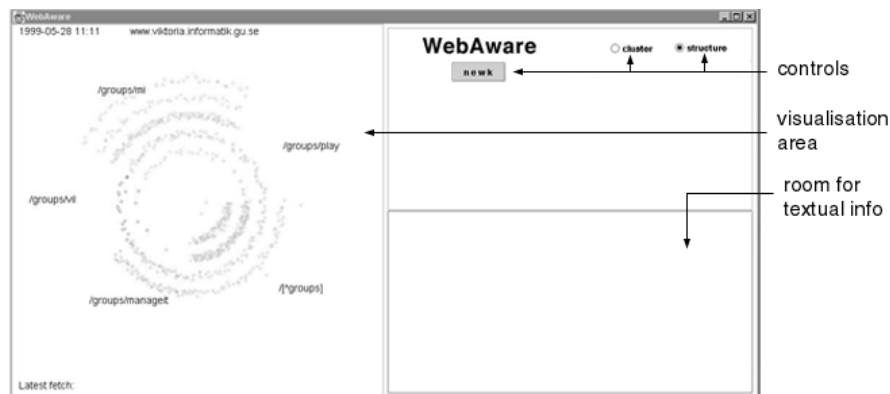


Figure 14: WebAware – screen version

4.5.2 Wall version

In the version meant for projection in a public space, no opportunity for user interaction is given, and so the only part needed of the above mentioned window is the actual map. Except for the window being enlarged, the map is the same as in the other version. Now, an obvious restraint is that only one map view can be projected at a time. There could be an automatic shift between the two at regular intervals, if this would not cause too much distraction. Investigations about this are still to be made.

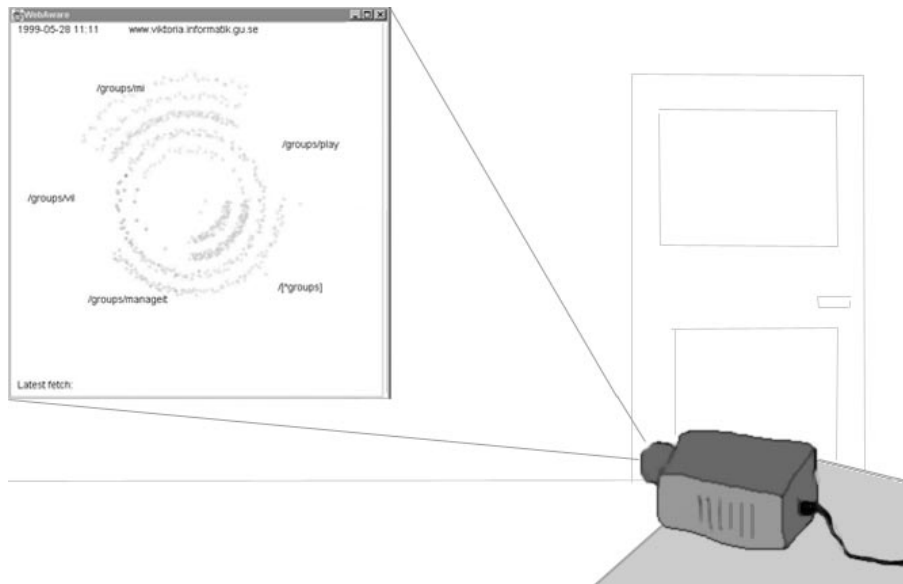


Figure 15: WebAware – wall version

5 Further work

Because of the time restraints of this project, there are of course limitations to the system. There are features that could be improved and others that could be added to make a more satisfactory product. There are also ideas which we knew all along would not be realised within our project, but which could hopefully give some inspiration for a possible further development of the system. We discuss these issues in this chapter, dividing it into the different areas in which we have found possibilities for improvement or expansion.

5.1 Tracking visitors

One feature of the system that we implemented only roughly and which we therefore decided to exclude in the final version was the dynamic drawing of paths for the most recent visitors. By this we could track a particular visitor and follow the steps in which he or she navigated the site. This is clearly a desirable feature, as it would help web designers to gain deeper knowledge about the visibility of links between pages or about what kind of information leads to what other stages in the navigation.

A related issue is that of the domains to which the visitors belong. It lies in the interest of the web site holder to know what kind of people the site attracts – for example if the user is connected to a university or to a company and from what part of the world the request is made. Since the IP number of the requesting computer is stored in the records of the log file, the only thing needed is a conversion from this number to the name of the computer. This is one of the simpler tasks we can think of in developing the system, but perhaps also one of the most interesting.

5.2 Clustering

The result of the clustering did not turn out to be what we expected, since one of the clusters contained roughly 90% of the pages. We have found a few possible reasons for this:

- *The categories are not representative for this domain.* We have not tried to categorise the pages with another set of categories and thus we cannot say whether or not the ones we have chosen to use are the ones that best reflect the contents of each page.
- *The contents of the majority of pages are in fact similar.* It could be the case that a great number of the pages have resembling contents and thus these pages have been merged into a cluster. When watching the

pages, however, you soon become convinced that this is a less likely theory than the one above.

- *The clustering algorithm is not as efficient as one would like it to be.* Again it is hard to say that it is so, since we have not actually tried any other methods for clustering.
- *The texts are too short.* Since many of the pages contain only a few lines of text, it is hard to adequately represent the contents of the pages with a vector. Since size does not matter to this method, the big and small pages will be compared as if they were alike.

As mentioned, we have used a relatively simple method, without thoroughly investigating the potential of alternative ones. If we would have done some more research, there is a chance we would have found a more appropriate one. However, even if we would have found a more elaborate method, there is a risk that it would not have been efficient enough to use in an application like this, where the processing speed is a major priority. Anyhow, research could be done and the most accurate and efficient method chosen.

In finding the categories to represent the documents to be grouped, hypernyms of WordNet fill the function satisfactorily. What could have been done in a different way is the selection of categories, which would benefit from being more representative according to what kind of a site is analysed. In this case, when the site belongs to a research institute and thus is of an academic character, semantic categories from the academic world should take up most of the positions in the vector.

A limitation in using WordNet is obvious when dealing with multilingual sites, since it is a database for English only. For a fully adequate analysis of a text, consideration has to be taken regarding in what language it is written, and then the text should be processed accordingly. The possible solution we can see is an automatic language guess followed by either a translation to English (for use of WordNet or other English reference system) or a lookup in one of a number of databases, one for each language used on the site.

5.3 Intelligent environment

The objective of WebAware to be a part of an intelligent and encalming technological environment is an idea that could be explored in much greater depth. There are numerous visions, and the Viktoria Institute seems to be the perfect breeding ground, since there are few limits there in what ideas can be considered rational or realistic.

In the WebStickers project performed within the research group PLAY [12] at Viktoria, the idea is to let bar codes represent web pages, so that when you scan the code, you are taken to the page in question. A future vision of the project is to have other devices, more of an everyday character than a browser, connected to bar code readers. An example of this in the case of WebAware would be that when a person or an object with a particular code is close to the display, this changes to show e.g. a closer view of the group of pages connected to that code.

An idea based on the relatively new concept of ambient media [13], and related to calm technology (see Section 2.2.5), is to let every person in the department select his or her own 'signature' – a little tune to be played every time a visitor requests his or her individual home page. This would certainly give a personal touch to the system and engage the interest of everyone in the department. However, at least two objections to this have to be admitted as relevant. The first one is that an application on a very popular site would cause quite a terrible noise, and so disturb rather than inform. The other one is that rivalry between staff members, that might have been provoked by the display of the site map, would be even more intensified...

5.4 Textual information

There are other possible connections to previous research on the Viktoria Institute, one of which could be to the OfficeID project [14]. It is a text generator using fragments of text processed within a computer network, giving a picture of what data are streaming through the network. This kind of text flow could be an interesting feature of WebAware as well, for example in showing a kind of summary of the information available on the pages most recently visited. This is something we would welcome as a future development, as it gives much additional information from data already collected for other purposes, namely for the content-based clustering (see Section 3.1.2).

The information on the web pages is not the only textual information we would have liked to add to the display. Also some explanatory text about the activity would be desirable, such as the number of visits, the URL:s of the most requested pages, the domains from which most requests come and similar facts. However, we would not wish to make too much of it, risking to draw attention from the graphic visualisation which is meant to be the main source of information. The only textual information we would like to add is the one that cannot readily be understood from the map.

6 Conclusion

Following the specification of the problem (see end of Section 2.2), let us investigate whether or not we have been able to achieve our goals with this system.

1. The system is to a great extent general and should - with a couple of small adjustments - be able to run on any web site using an Apache web server. However, for really vast sites, it might be a good idea to modify the metaphor in some way, e.g. by making it three-dimensional or by letting one icon represent several pages.
2. By grouping and clustering the pages, we have made obvious what pages are related to each other (in either structural or semantic terms) and by labeling these groups and clusters, we have shown in what way they are related.
3. Since the system runs in real time, the dynamic effect is highly dependent on the amount of traffic on the site. In this way we find that the system paints a true picture of what is going on. “You don’t think the visualisation is dynamic enough? Make your site more interesting and you’ll see dynamics.”

As mentioned in the previous chapter, the system could be improved in many respects, but the major goals have still been achieved.

The WebAware project has resulted in a web statistics system a little different from the ones on the market. The fact that it runs in real time is only one of the features that separate it from most of the ones we have encountered. This gives it the status of a live performance as opposed to a prerecorded one. The geographic picture of the site is another feature that is not at all universal for systems of this kind. The third feature we would like to emphasise as a distinguishing factor is the categorisation of pages for the purpose of a semantically organised map. Although this is an area where great improvement could be made, the attempt to give a content-based structure to the display is interesting as a first prototype. It should not be denied that the semantic ingredient adds an extra flavour to the system.

References

- [1] *WebTrends Log Analyzer* Copyright 1995-1999 WebTrends Corporation, Portland, Oregon.
<http://www.webtrends.com/products/log/default.htm>
- [2] *The fishFinder Applet*
<http://developer.javasoft.com/developer/FishFinder.html>
- [3] Minar, Nelson. *Visualizing the Crowds at a Web Site* In Extended abstracts of CHI'99, Pittsburgh, Pennsylvania, USA, May 1999.
<http://nelson.www.media.mit.edu/people/nelson/research/crowdvis/>
- [4] Dewe, Johan. *En prototyp för att klassificera dokument från WWW med avseende på genre och ämne (A Prototype for Classification of Documents from WWW, with regard to Genre and Content)* January 1998.
<http://www.student.nada.kth.se/~d92-jde/examensarbete/DropJaw.html>
- [5] Holmquist, Lars Erik et al. *Navigating Cyberspace with CyberGeo Maps* In Proceedings of IRIS 21, Saeby, Denmark, 1998.
<http://www.viktoria.informatik.gu.se/publications/98/VRR-98-10.pdf>
- [6] Card, Stuart et al. *Readings in Information Visualization; Using Vision to Think* Morgan Kaufmann Publishers, January 1999.
- [7] Rohrer, Randall M. and Swing, Edward. *Web-Based Information Visualization* In IEEE Computer Graphics and Applications, July/August 1997.
- [8] Weiser, Mark and Brown, John Seely. *Designing Calm Technology* In Powergrid Journal 1.01.
<http://electriciti.com/1.01/calmtech-wp.html>
- [9] Miller, George A. et al. *Introduction to WordNet: An On-line Lexical Database* Revised August 1993.
<http://www.cogsci.princeton.edu/~wn/>
- [10] Josefsson, Martin and Rasmark, Torgny. *CO-PRO – The use of CONcept PROfile vectors in domain-sensitive content comparison of text documents* Forthcoming M.A. thesis in Computational Linguistics, Göteborg University.
<http://www.cling.gu.se/~cl4trasm/Omhet/index.html>

- [11] Jacobs, Ian and Raggett, Dave. *HyperText Markup Language Home Page* May 1999
<http://www.w3.org/MarkUp/>
- [12] *The PLAY Research Group*
<http://www.viktoriamatik.gu.se/groups/play/>
- [13] Ishii, Hiroshi and Ullmer, Brygg. *Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms* In Proceedings of CHI'97, Atlanta, Georgia, USA, 1997.
<http://www.acm.org/sigchi/chi97/proceedings/paper/hi.htm>
- [14] Jaksetic, Patricija and Redström, Johan. *OfficeID – Generating Texts Using Dynamic Information in a Computer Network* In Proceedings of Writing and Computers '99 (poster), Cambridge, UK, March 1999.
<http://www.viktoriamatik.gu.se/~johan/abstracts/aofficeid.html>

A Appendix

The following sections show an example of a web page, its source code, the text extracted from it (including the result from the count of links, images and e-mail addresses) and finally the resulting vector.

A.1 Web page of Mobile Informatics

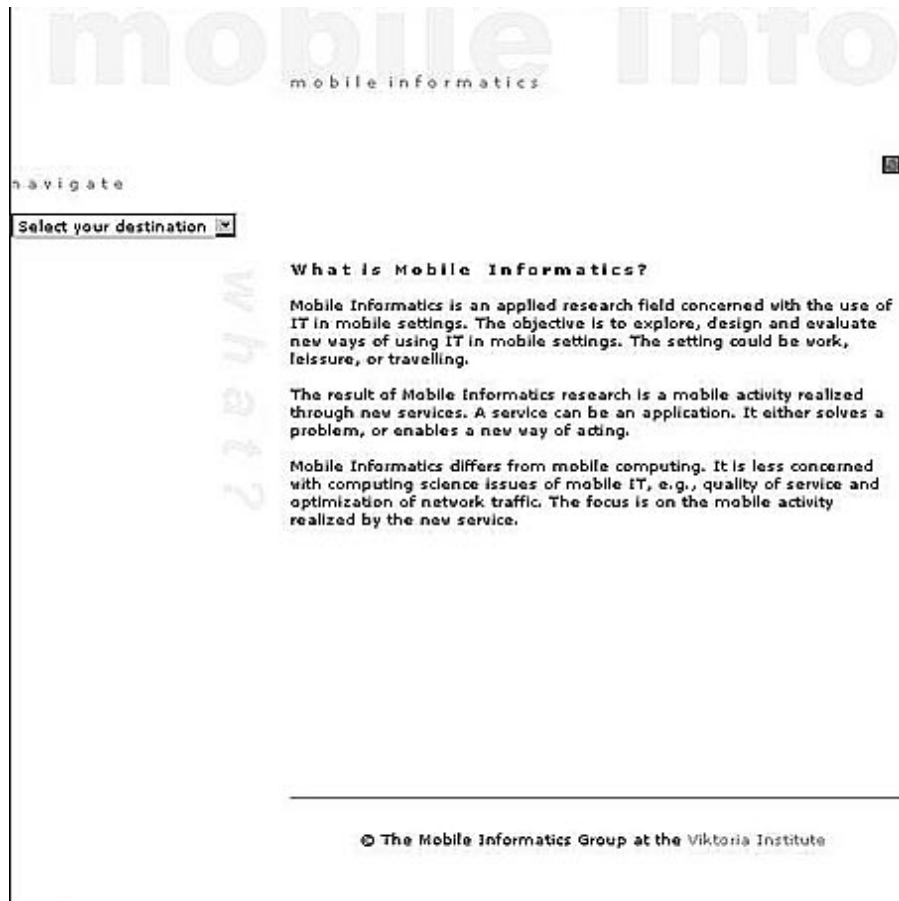


Figure 16: Example of a page on the Viktoria site

A.3 Extracted text

0 0 3 3 4

m O b i l e i n f O r m a t i c s m o b i l e i n f o r m a t i c s n a v i g a t e
Select your destination Start What How The problem Results Publications
Concepts Services Resources Projects Cometa Conny MobiLinq MobiNews
MobiService MobiCom Pendulum Overload Scenario Events Seminars People
Contact person Search W h a t i s M o b i l e I n f o r m a t i c s Mobile
Informatics is an applied research field concerned with the use of IT in mobile
settings The objective is to explore design and evaluate new ways of using
IT in mobile settings The setting could be work leissure or travelling The
result of Mobile Informatics research is a mobile activity realized through new
services A service can be an application It either solves a problem or enables a
new way of acting Mobile Informatics differs from mobile computing It is less
concerned with computing science issues of mobile IT quality of service and
optimization of network traffic The focus is on the mobile activity realized
by the new service The Mobile Informatics Group at the Viktoria Institute

A.4 Vector file

0 0 3 3 4 10 10 10 4 101 4 27 17 115 30 7 10 5 10 4 10 2 11 8 36 1 5 0.0001 1
36 0.0001 1 79 7 7 8 1 19 30 10 1 4 1 3 2 24 7 6 7 44 10 20 3 13 1 16 5 2 5
7 6 7 2 1 12 0.0001 8 1 5 7 7 3 2 14 10 2 0.0001 63 1 0.0001 1 12 1 10 3 28
7 90 4 7 6 7 0.0001 0.0001 1 1 14 3 18 8 17 6 1 1 14 2 5 3 4 32 1 5 4 0.0001
5 7 2 0.0001 2 1 1 0.0001 0.0001 2 5 5 25 18 4 3 6 5 3 2 13 6 18 18 19 13 23
11 0.0001 2 8 67 8 10 16 25 10 0.0001 8 29 4 10 1 5 8 55 17 90 6 0.0001 7 1
0.0001 21 3 16 5 6 2 10 13 20 21 4 16 6 0.0001 3 3 12 20 1 12 3 4 14 29 1 2
1 1 1 6 10 10 10 14 20 10 1 4 4 8 14 10 1 0.0001 0.0001 23 1 2 10 5 55 1 3 3
10 7 7 1 4 7 2 5 1 6 4 11 3 5 47 0.0001 0.0001 10 24 9 2 4 6 10 17 1 35 3 2
0.0001 1 1 1 groups+mi3+what+index.txt