

*Intelligent user interface
for specialized web sites*



Edouard Forler

Diploma thesis 1999/2000

Artificial Intelligence Laboratory - Computer Science Department
Federal Institute of Technology Lausanne

Director
Pr. Boi Faltings

Supervisor
Dr. Martin Rajman

Keywords: Dynamic web interface, e-commerce, cybermarketing, user-adaptive system.

With the recent exponential growth of the internet, companies have been having the opportunity to discover always wider and more potential markets. Concurrently to these markets becoming planet-wide, some demand has appeared for more proximity to customers and care for their precise needs.

Such paradoxal requirements cannot be fulfilled by hand only. They are need to be handled by automated systems able to browse a wide variety of products and services to find the best answer to a customer request. But such systems, while being extremely powerful at storing, sorting or even associating informations together, still lack the adaptivity and flexibility of a human salesman, essential qualities when dealing with customer satisfaction.

This report presents some ideas and techniques that can be used to implement such qualities. In particular, we have adapted some algorithms traditionnally used to accelerate navigation through a hierarchy, in order to implement some basic anticipation of user wishes. The resulting mechanism has been integrated within a web server interacting with a real-life medium-sized product and service database.

Mots-clefs: Interface web dynamique, e-commerce, cybermarketing, système adaptatif.

Ces dernières années, avec l'essor incroyable de l'internet, les entreprises ont découvert les moyens de toucher des marchés toujours plus étendus et prometteurs. Dans le même temps, une demande de proximité et d'attention plus importantes s'est faite sentir de la part du client.

De telles exigences paradoxales ne peuvent être remplies manuellement. Elles doivent être prises en charge par des systèmes informatiques, les seuls capables de rechercher, parmi plusieurs millions de produits et de services, celui correspondant à la demande du client. Cependant, bien que très doués pour le stockage, le tri et l'association d'information, ces systèmes ont le défaut majeur de manquer de souplesse et d'agilité, qualités indispensables lorsqu'il s'agit de satisfaire le client.

Ce rapport présente quelques idées et techniques qui peuvent être employées pour approcher de telles qualités. Nous avons en particulier modifié des algorithmes servant habituellement à accélérer le parcours de hiérarchies, dans le but d'implémenter une forme rudimentaire d'anticipation des désirs de l'utilisateur. Le mécanisme résultant a été intégré à un serveur web fonctionnant en interaction étroite avec une base de données concrète de les produits et de services.

List of Figures	I
1 Introduction	1
1.1 Goals	1
1.2 Document structure	2
1.3 Administrativia	2
1.4 Acknowledgments	2
2 Key concepts	3
2.1 Basic idea	3
2.2 The IOI algorithm	4
2.3 Our proposition	6
2.3.1 Score distribution	6
2.3.2 Extending documentary searches	7
2.3.3 Sorting	8
2.3.4 Lattices considerations	10
2.4 Actual calculation techniques	12
2.4.1 Measuring user interest	12
2.4.2 Handling user documentary requests	13
2.5 Situations where the algorithm can help	15
3 Implementation	17
3.1 Overall structure	17
3.2 Server side	18
3.3 Client side	18
4 Results	20
5 Conclusion	22
A Database architecture	23
Bibliography	26

List of Figures

2.1	A simple categorization example	3
2.2	Different approaches in categorizing 10 targets	4
2.3	Step one: scoring	5
2.4	Step two: propagation	5
2.5	Step three: downward traversal	6
2.6	Worst case with a restricted distribution	8
2.7	Standard result display	9
2.8	Results ordered by their score, best first	9
2.9	Examples of latticed classification	10
2.10	Propagating scores in a latticed structure.	11
2.11	Early stopping in a lattice.	11
2.12	Keeping track of the source	12
2.13	Keywords shared among the nodes.	14
2.14	Problems with Dice and Jaccard's coefficients	15
3.1	Overall structure	17
3.2	Tracking mechanism	19
A.1	Raw data from which the structure is built	24
A.2	The resulting classification	25

CHAPTER 1

Introduction

My way is to begin with the beginning.
Lord Byron - Don Juan, c. 1, dedication vii.

1.1 Goals

E-commerce is something that up-to-date companies cannot ignore. It is more than just a matter of being in the vogue. Every day, we can see how the internet phenomenon and computing in general are revolutionizing our way of life. Buying things on the net is so easy and much more comfortable than going shopping when the sales are on.

Or perhaps not. Billions of products produced by millions of different companies to choose from. More innovation every week. Out of date and sometimes inaccurate information. E-commerce is not yet what it is meant to be, that is, a fast, easy and reliable mean of trade.

Moreover, traditional commerce allows for more conviviality. You can take your time to explain what you're looking for. Sales persons usually do their best to try and satisfy your wishes. They often offer a smiling face and will even sometimes choose for you or give you advice for a better purchase.

These are essential qualities that even the best E-commerce systems obviously lack dramatically. Of course, our goal here is not to create the perfect robotic salesman in the Frankenstein manner (or perhaps more friendly) , since most of the qualities we are talking about do not have an equivalent in the virtual world (or we simply do not know how to implement them).

Still, one aspect deserves careful study, because clues exist about how to transpose it into computer algorithms. This aspect is the capacity for a well-enough informed person to give some advice to a customer for a better purchase.

We believe that it is possible to build an algorithm that will be able, by analyzing customer behaviour and requests, and comparing this information to what is known about available products, to better answer queries and infer trends and fashions, thus behaving as the real salesman: “You should try this one, it’s the latest fashion” or “I think I know what would suit you” are well-known sentences in real commerce.

1.2 Document structure

Chapter two will introduce an algorithm traditionnally used to accelerate navigation through hierarchies and show how we adapted it in order to achieve our goals.

Chapter three will focus on keys for implementation within a real web server, both on the server side and on the client side, and some ways to build what we call a user profile.

Chapter four will summarize the main results obtained during the system implementation.

Chapter four will give the conclusion to this work.

1.3 Administrativa

This work has been conducted within the Artificial Intelligence Laboratory (LIA) at the Swiss Federal Institute of Technology in Lausanne during winter 1999/2000 and presented as a diploma thesis in order to obtain the grade of Computer Science Engineer.

The project was directed by Pr. Boi Faltings and supervised by Dr. Martin Rajman.

1.4 Acknowledgments

Many thanks go to my supervisor, Dr. Martin Rajman, for his precious help with graph theory and complexity calculations, both of which being indispensable to prove the feasibility of the developped mechanisms.

‘Take some more tea,’ the March Hare said to Alice, very earnestly.
 ‘I’ve had nothing yet,’ Alice replied in an offended tone, ‘so I can’t take more.’
 ‘You mean you can’t take *less*,’ said the Hatter: ‘it’s very easy to take *more* than nothing.’
Lewis Carroll - Alice in Wonderland, ch 7.

2.1 Basic idea

The idea is to ease user navigation through a web site proposing a large amount of products and services in various categories. Existing solutions to the problem of browsing so much information are usually to sort the available products and services¹ in categories and sub-categories and allow the user either to browse the categories or to request for a particular product by giving a description in the form of a set of keywords (Fig. 2.1).

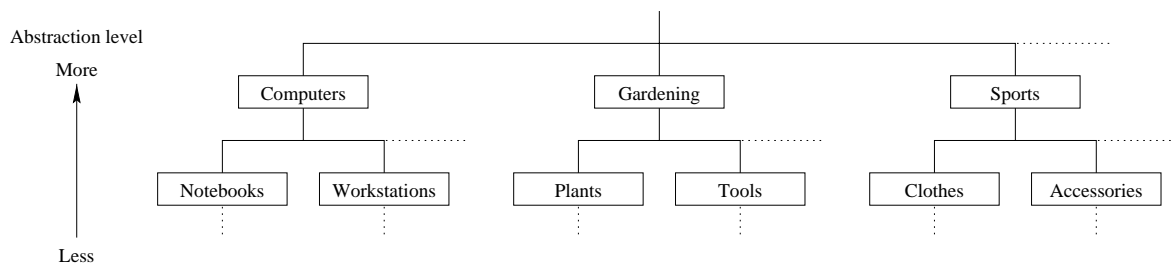


Figure 2.1: A simple categorization example

Such implementations are widely used nowadays, starting with the very popular Yahoo![®] search engine. But this technique, while efficient with limited numbers of nodes and targets in the hierarchy, has shown to be quite ill-adapted to large amounts of information:

¹Products and services will from now on be called *targets*, as opposed to *nodes*, referring to the (sub)categories in which the products and services were classified.

- Either the results found by the system when entering a subcategory are too numerous, and the user often does not know how to determine his next move;
- or the number of different available items in a subcategory is artificially kept low, but this means that the deepness of the hierarchy must be increased. The visible result is that the user has to traverse more successive nodes to reach a target (Fig. 2.2).

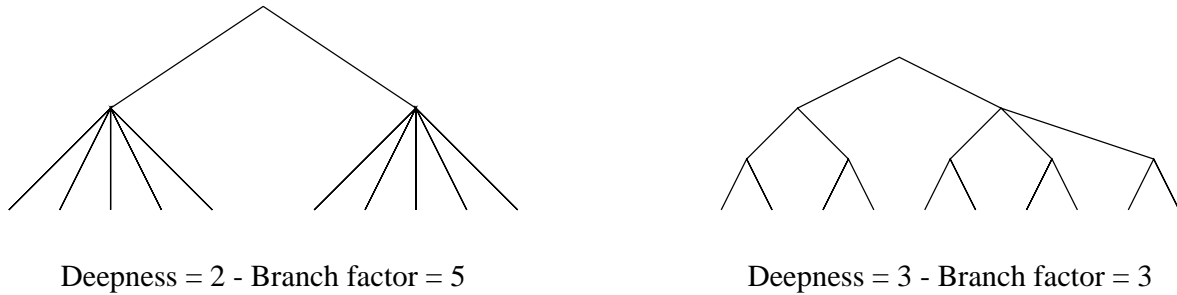


Figure 2.2: Different approaches in categorizing 10 targets

These problems are inherent to the tree structure and cannot be avoided when dealing with raw trees. In both cases, the user will waste some time, either because he has to think which node to choose among many proposed, or because he will have to browse through a deeper hierarchy.

The method we have implemented is a variant of the IOI navigation algorithm proposed by Uwe Gühl [1] and Thomas Wolters [2] for the design of natural language based interfaces and accelerated help systems.

2.2 The IOI algorithm

First, we need our classification tree to be very deep and a branch factor as reduced as possible²: studies have shown that a normal human being is able to directly perceive up to three different objects without any thinking process, and that for up to five to seven objects, the cognitive (in fact counting) process is very short (short term memory). So building a deeper tree will shorten the user's decision time, but increase the overall length of the traversal and therefore the overall navigation time. The latter problem can be solve with the IOI algorithm³.

The idea behind the IOI algorithm is to compute a score distribution on the targets in the tree and to propagate the best scores through the tree up to the root. Downward navigation can then rely on the branches with the best scores. Choices at the nodes are made automatically until the difference between two paths' scores is lower than a predefined threshold⁴.

The score distribution can be, for example, based on a query (pertinence of keywords) given by the user to the system, according to textual descriptions associated with the targets.

²Methods about how to build an adequate target classification will not be discussed here.

³We will from now on use this term to designate previous works, referring to [1] and [2].

⁴Attempts at estimating it are introduced in [3], chap 3.

This algorithm will thus determine the most relevant node in the tree, according to the user's request. The system can then automatically lead the user to this node, skipping all previous nodes in the hierarchy, since the user would have probably choose them if he had had to explicitly search the tree "by hand". The technique can be split in three steps:

Score distribution

According to a pertinence formula, a score is computed for each of the target (Fig. 2.3).

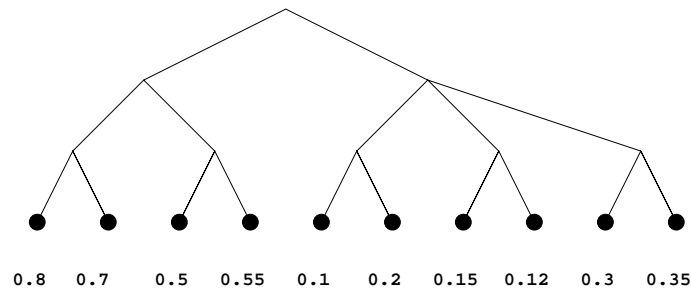


Figure 2.3: Step one: scoring

Upward propagation

When the score computation is complete, the algorithm selects the best scores at each level and propagates them upward until the root is reached (Fig. 2.4).

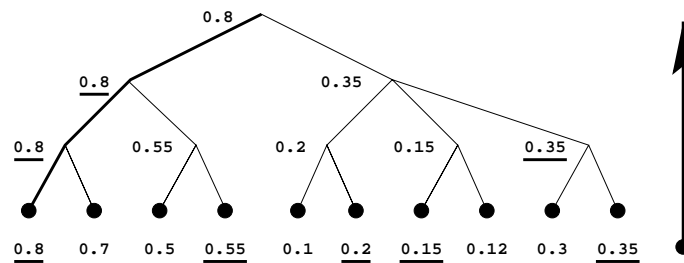


Figure 2.4: Step two: propagation

Automatic downward traversal

The best score now stands on top of the tree. It is used as a marker in the last step of the algorithm to automatically traverse the tree downward. When the difference between the scores of two children nodes is below a certain threshold, the process is stopped and the resulting node will be the next starting position in the tree.

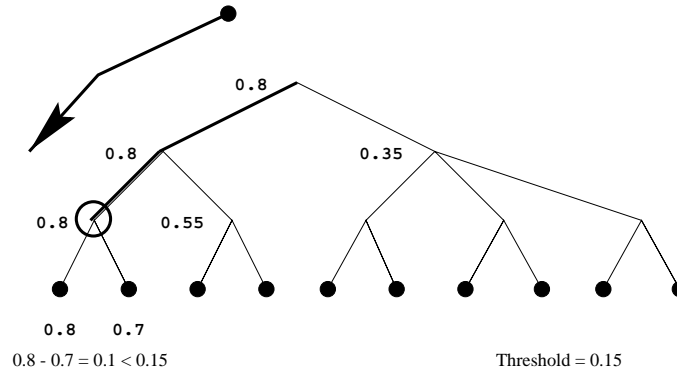


Figure 2.5: Step three: downward traversal

Of course, this navigation algorithm can be successfully applied to a subtree, that is, we can assimilate the root of the tree presented in fig. 2.3 to any node of a bigger tree; For example, the user may have made a request that led him to this node, and is now submitting a more precise query producing the result shown earlier.

2.3 Our proposition

The modifications we have made to the initial IOI algorithm are the following:

1. The score distribution is computed on the basis of not only user queries, but also of the interest that the user has shown for each target during previous interactions with the system;
2. the documentary search can be applied to the entire tree or only to a subtree, and results are compared together;
3. results are shown sorted by their relevance, best first;
4. the resulting algorithm has been enhanced to be able to manage lattices instead of simple trees.

2.3.1 Score distribution

Let's suppose we have built a tree with n targets. Each time a user u makes a move in the structure, either because he has chosen a subcategory in a list, or because he has produced a query, the system computes a discrete score distribution on all the targets. For each target, the score s_j is calculated according to the following formula:

$$s_j = \frac{w_u \bar{v}_u + w_g \bar{v}_g + w_s f_s(D, d_j)}{w_u + w_g + w_s}, 1 \leq j \leq n$$

where \bar{v}_u is the interest shown by u during previous interactions and \bar{v}_g is what we call the general interest, that is, the mean of the interests shown by all known users during previous interactions. We have the following formulae:

$$\bar{v}_u = \frac{\sum_{k=1}^m i_k}{m} \text{ and } \bar{v}_g = \frac{\sum_{k=1}^{n_u} \bar{v}_{uk}}{n_u}$$

with m the number of interactions per user, i_k the resulting interest from a single interaction and n_u the number of users.

Finally, $f_s(D, d_j)$ is a documentary score which estimates the relevance of a textual description d_j of the target j by comparing it to the query D expressed by the user. Different scoring methods exist with various results (see section 2.4.2).

w_u , w_g and w_s are weights which allow us to tune the behaviour of the system. Thus, if we want our algorithm to be very user-aware, we can set w_u to a high value while keeping w_g and w_s low; or we can decide that our users should be influenced by the latest fashion and we set w_g to a higher value and so on. Of course care must be taken that w_s has to be temporarily set to nil when the user is just making a move in the structure without submitting any query.

As mentioned earlier, the scoring formula is still valid when applied to only a smaller part of the tree, keeping in mind that n becomes the number of targets that can be reached from the top node of the subtree only: the other targets are products that are linked to categories which the user has decided not to browse anyway⁵.

2.3.2 Extending documentary searches

It has been found that, when a user submits a query, restricting the score calculation to targets linked to the current node of the tree is too constraining; of course, once the user has reached a node that matches his request, he may want to refine the search by giving a more detailed description of the target he is seeking.

But the system might also have led him to something that is not actually suitable. In the latter case, the user will probably try to escape his current portion in the tree, either by going up the hierarchy, or by submitting a new textual description. But in this case, the new description has little chance of succeeding if the calculation is limited to targets linked to the current node (Fig. 2.6).

⁵Simply cutting off irrelevant targets may be a bit ingenuous in fact; see sect. 2.3.2 and 2.3.4.

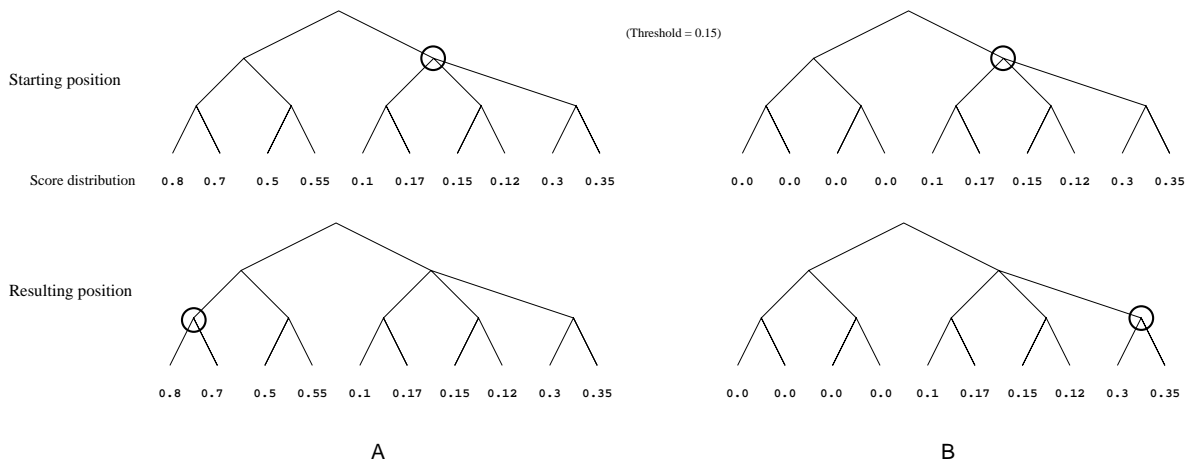


Figure 2.6: The worst case with a restricted distribution: in (B), the system consolidates the user’s mistake by omitting targets that would have given far better results.

In the current implementation of the system, the user is given the possibility to choose between “a search in the current category” and “an overall search”, which eliminates any ambiguity. But this may become laborious on the user’s side, so it would be advisable to design a robust mechanism aimed at jumping through the hierarchy, taking into account or not the user’s judgment (See [3], chap. 4).

2.3.3 Sorting

Once the system has found a node that fits, it displays the children nodes (subcategories) linked to the node it has determined, so the user can make up his mind about his next move.

For example⁶, let’s consider a user looking for a sound card for his computer. After the initial query (“I want a sound card”), the system shows the node “Sound cards” (Fig. 2.7):

⁶Further discussions are based on a fake product database detailed in appendix A.

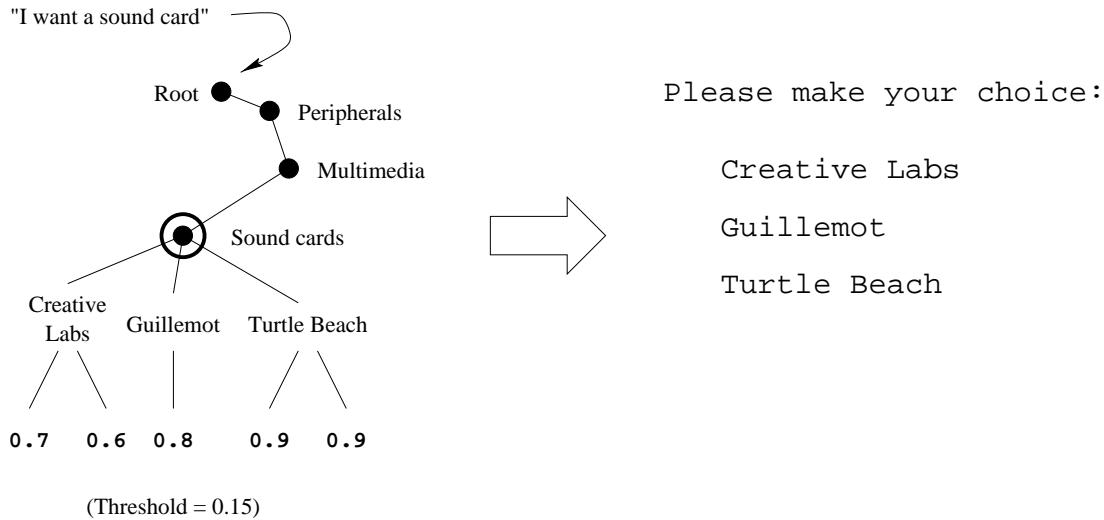


Figure 2.7: Standard result display

Now, let's imagine that the same user has visited several times the hierarchy, got particularly fond of Turtle Beach products and that he is returning periodically to check the prices. We may reorder the results so that they appear as in fig. 2.8 because we know that the user has shown more interest for these products in the past (but still not enough to discriminate clearly Turtle Beach from other products in the same category).

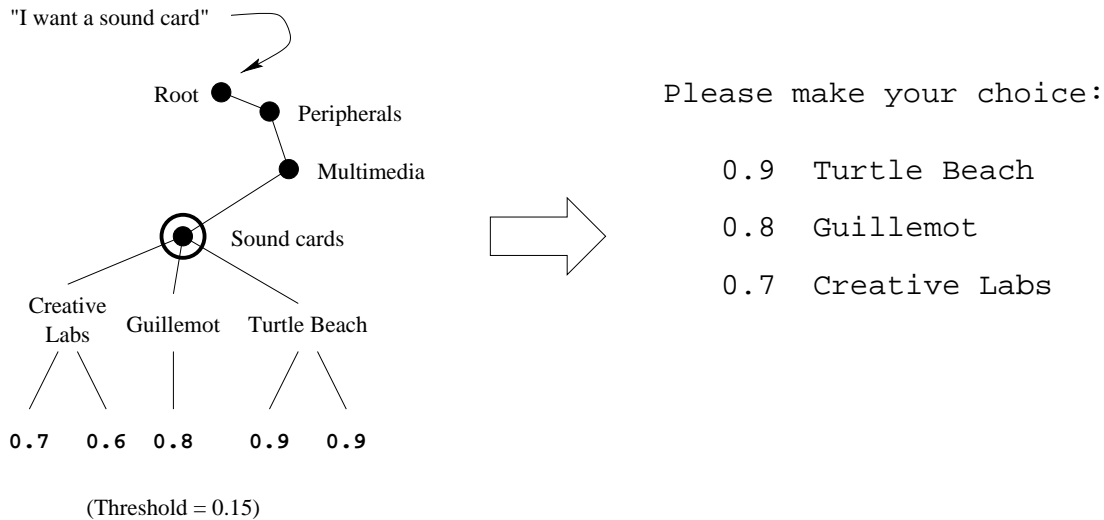


Figure 2.8: Results ordered by their score, best first

By adding this feature, we have managed to put up a system that is able to find the best products for a customer, because it *knows* both of his habits and of the products supplied, and hopefully to help him find what he is seeking more efficiently and more rapidly.

2.3.4 Lattices considerations

One big problem that appears when dealing with queries is that there are many ways to express one's request. The keyword list describing a target is unlikely to be exhaustive. Moreover, one's description may be rather different depending on what one has in mind. For example, a piano will obviously be described as a "musical instrument" by a pianist, whereas a removal man may rather classify it as "furniture"; a system administrator will classify a CD-ROM drive as "mass storage" but any gamer will tell you it is a "multimedia device".

The solution here lies in the extension of our tree structure in a way that any node can be linked to more than one parent, that is, to extent the structure to a semi-lattice.

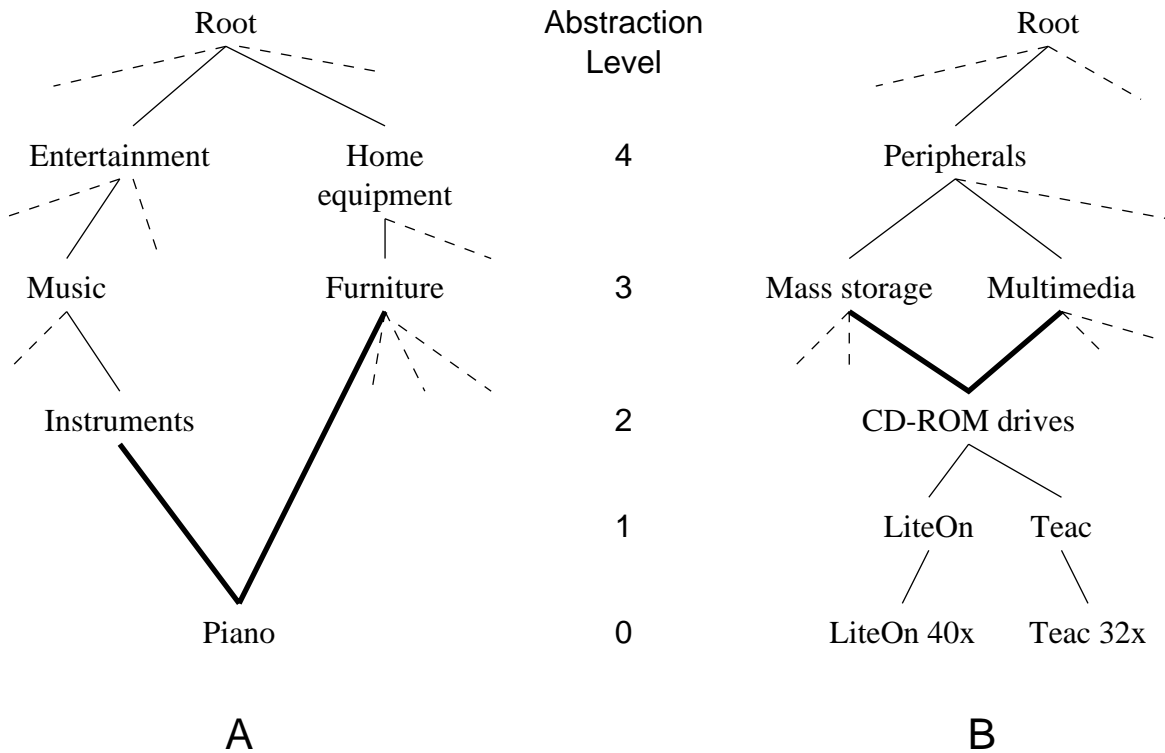


Figure 2.9: Two examples of latticed classification: (A) a target and (B) a node. Skipping levels is allowed as well (A).

Using lattices provides a convenient way to define "shortcuts" in the hierarchy when needed and eases the classification of ambiguous targets (Fig. 2.9).

Let's have a closer look at how the algorithm behaves when applied to a semi-lattice. When the scores are propagated upward, nothing opposes to duplicating the values along the multiple branches a node is connected by; going over more than one level at a time is not a problem either (Fig. 2.10).

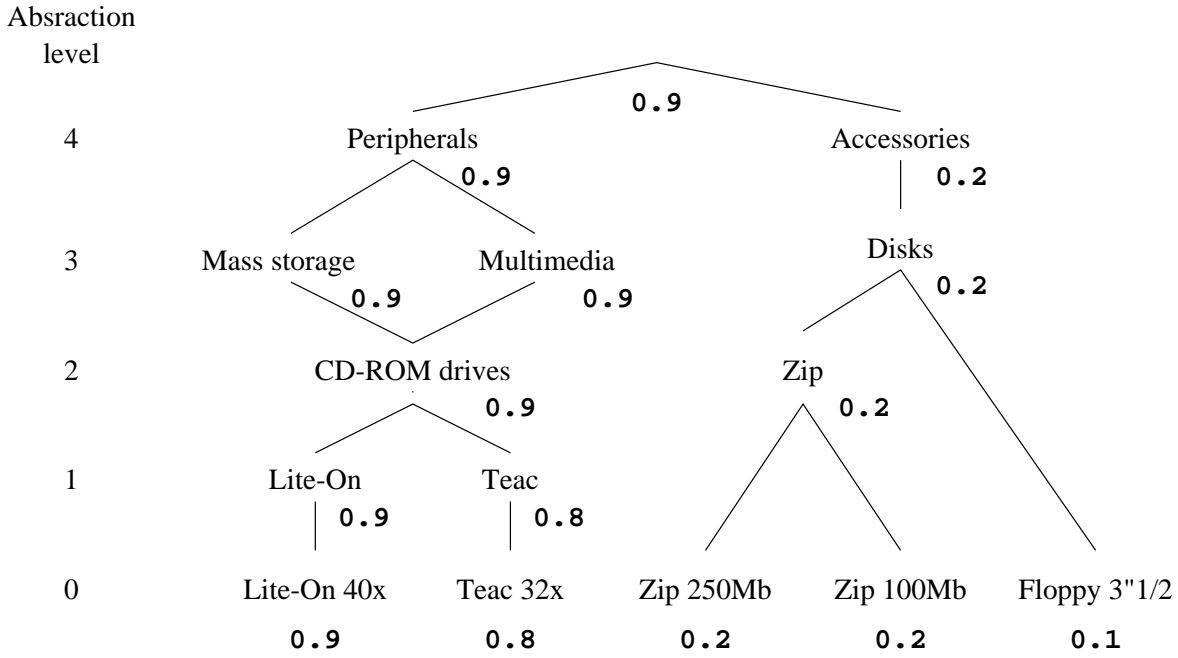


Figure 2.10: Propagating scores in a latticed structure.

Unfortunately, we can clearly see the problem with the last step of the algorithm: it gets stuck on nodes that are starting points for more than one path to the same target, because it sees multiple targets with the same score where there is in fact only one. This creates an ambiguous situation that cannot be dealt with (Fig. 2.11).

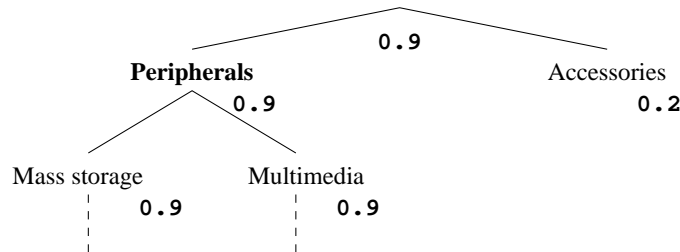


Figure 2.11: Ambiguous situations make the algorithm stop too early (here on “peripherals”).

How can we clearly state that there is only one target to be considered? The easiest way to clarify the situation is to keep track of the source of the propagated scores. This can be done by identifying targets with linear coordinates that will be propagated concurrently to the scores, thus acting as signposts during downward traversal (Fig. 2.12). In case of an ambiguity during this process, the algorithm will arbitrarily choose the first path leading to the target ⁷.

⁷A clever implementation would choose the shortest one. See [3], chap. 4.

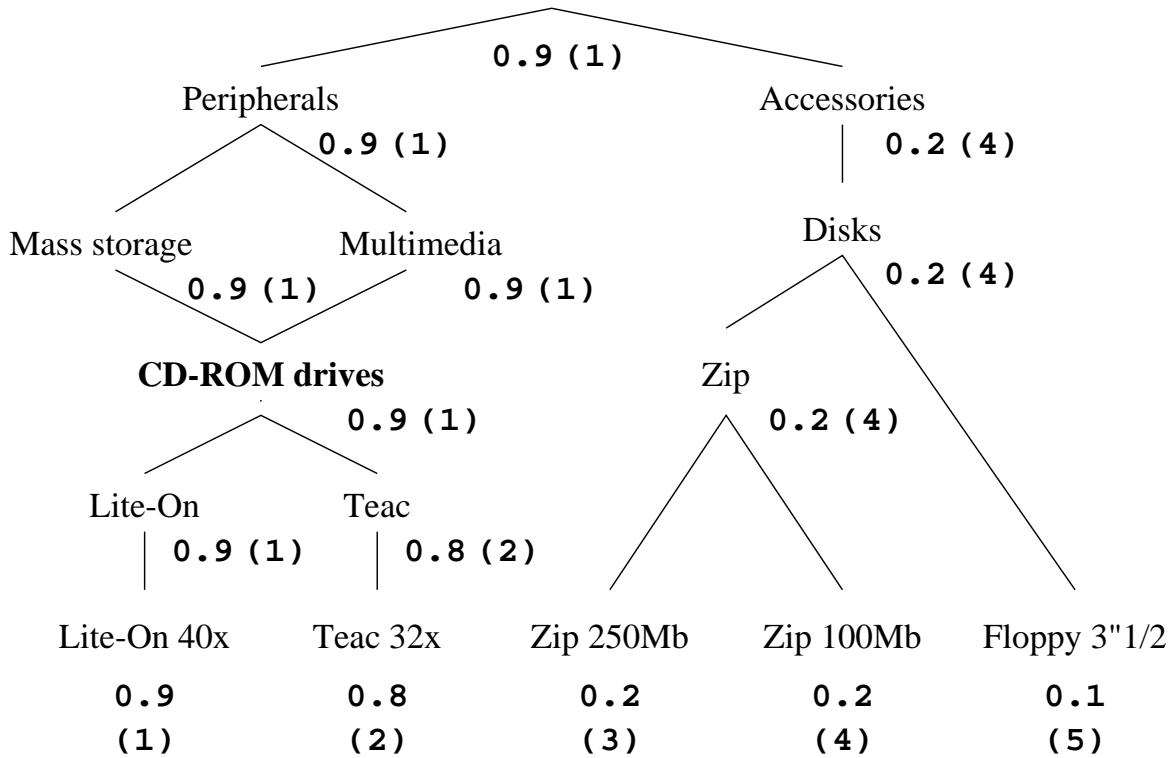


Figure 2.12: Keeping track of the source, the algorithm stops on the node “CD-ROM drives”, which is the expected result.

2.4 Actual calculation techniques

2.4.1 Measuring user interest

All the discussion till now is based on the hypothesis that the system is able to determine as fairly as possible the interest a user has shown for the different targets during previous interactions.

In our experimental implementation, this was achieved by asking the users to give an appreciation each time they found a target. Five levels of appreciation were proposed:

- “This product doesn’t match your request at all”;
- “This product doesn’t match your request very well”;
- “This product matches quite well”;
- “This product matches perfectly”;
- “You don’t know whether this product matches your request or not”.

The last possibility is neutral and should be chosen in case the product is not known by the user or if he does not understand the link between his request and the result.

To be able to go on browsing, the user was compelled to answer the question and could then choose between looking for a better match to his request, starting over for another product or ending his session.

After the user has logged out, the marks are stored and converted to decimal values ranging from 0 to 1, and can be directly used in our score formula (see sect. 2.3.1). While constraining, this method insures reasonably good results with the interest distribution on the targets.

Of course, real life teaches us that shops hardly ever ask their customers to fill such questionnaires, at least not every time they buy something. But a salesman usually guesses whether a customer will return or not, since his behaviour may reveal his thoughts. Such indications can be somewhat transposed to e-commerce and replace the need for a questionnaire. These techniques are discussed further in [3], chap 4.

2.4.2 Handling user documentary requests

Building pertinent descriptions

As stated in section 2.3.4, building an exhaustive list of keywords describing a target is a very long and tough process. To simplify the task, we propose to associate keywords not only to the targets but also to the nodes in the hierarchy and to choose very specific keywords. It is easier to think of keywords directly related to an object than to try to draw up a list containing also keywords referring to the overall category which the object belongs to. Moreover, since a target can be attached to more than one parent, forgetting keywords becomes a real concern. Attaching specific keywords to the nodes and gathering them on the targets greatly eases the description process when dealing with thousands of targets.

In our approach, keywords describing the nodes are not directly used by the search algorithm, but gathered together to create a (hopefully almost exhaustive) descriptive list for the target (fig. 2.13).

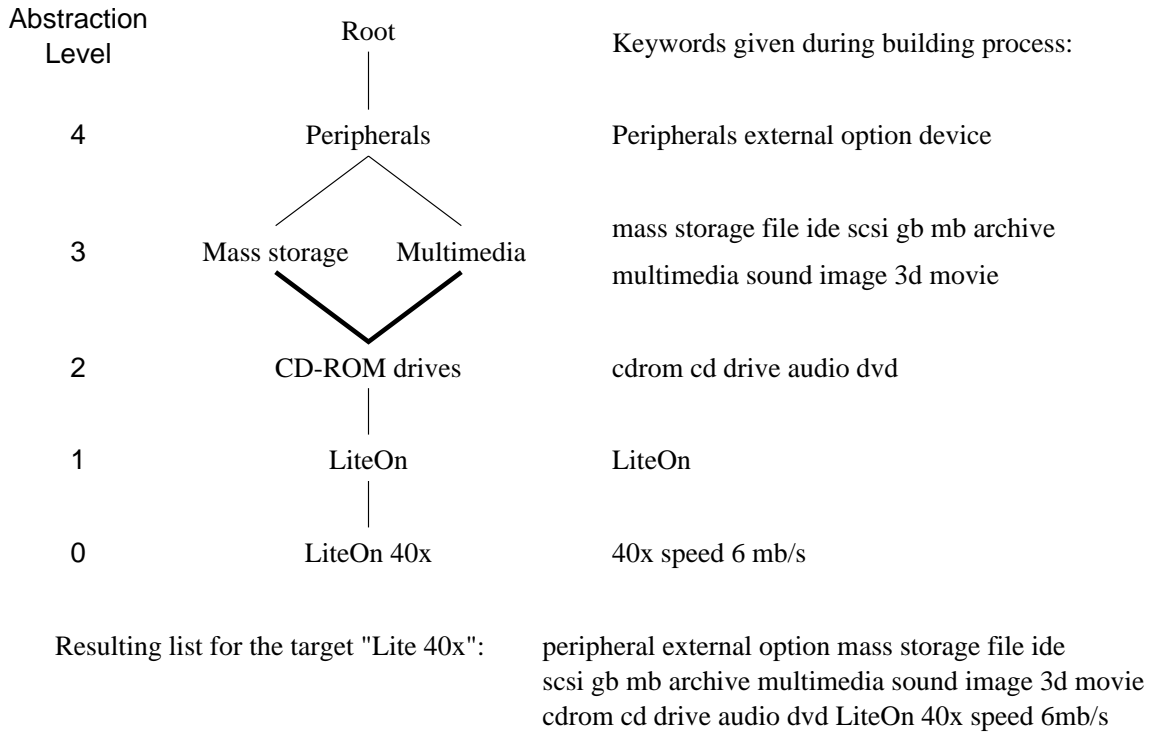


Figure 2.13: Keyword shared among the nodes are gathered together to build lists describing targets.

A deeper hierarchy results in a better description of targets since it is possible to refine descriptions for the nodes at each abstraction level.

Exploiting descriptions

Any computer scientist can tell how difficult it is to find a reliable method to estimate the resemblance between two textual descriptions. Since this is not the main subject of our research, we decided to implement simple and well-known approaches. The first attempt with implementing documentary searches was to compute Dice and Jaccard's coefficients:

$$Dice = 2 \frac{|D_1 \cap D_2|}{|D_1| + |D_2|} \qquad Jaccard = \frac{|D_1 \cap D_2|}{|D_1 \cup D_2|}$$

which give an estimation of resemblance between two texts D_1 and D_2 by computing the proportion of words being present in the two documents.

Unfortunately, we quickly discovered that such coefficients were completely inadequate; this problem comes from the descriptions associated to each target, which are usually short (about five to fifteen words). Dice and Jaccard's coefficients are heavily influenced by the overall number of words in the documents, since they are nothing else but a ratio between the number of shared words and the total number of words. Because our descriptions are short and user queries are even shorter (usually between two and five words), we sometimes get pretty surprising results (Fig. 2.14).

		Dice	Jaccard
"CD-ROM drive"	vs "cd-rom cd movie music drive multimedia audio dvd 40x mass storage"	0.307	0.181
"CD-ROM drive"	vs "zip drive mass storage"	0.333	0.2

Figure 2.14: And yet I was asking for a CD-ROM drive...

To avoid this problem, the easier way would be to make sure that all the descriptions have the same length. Since this is a very restricting condition (There may be too many or too few words available to describe a target), we decided to suppress the dependence to the length of the descriptive document by implementing a modified version of Jaccard's coefficient:

$$\text{Modified Jaccard} = \frac{|D_1 \cap D_2|}{|D_1|}$$

While solving our main problem here, this formula however suffers from a lack of precision: because the user's query is usually short, that is about two to three relevant words, the score distribution is limited to few different values; let's have a three-word long query and a hundred targets, and the possible scores will be 0.0, 0.33, 0.67 and 1.0, thus grouping tens of targets together even if the correlation is relatively weak.

It is clear though that the main goal of this project is not to provide a good documentary search method. Nevertheless, we think it would be interesting to complete the system with a more robust search algorithm which could include query expansion or even more subtle analyses (grammatical or semantic, and so on) of the user's queries, since good results in this part would lead to better customer satisfaction and both help to validate the classification and strengthen data acquisition.

2.5 Situations where the algorithm can help

Regular customers

This is the situation where a customer is returning regularly to check prices, availability, or because the target he is looking for changes constantly. For example, let's imagine a website specialized in Stock Exchange. The first thing shareholders usually do is check their portfolio. In such a situation, the algorithm would take a regular customer directly to rates, and could even propose, if the structure of the hierarchy is cleverly built, a list of the shares the customer usually checks. On traditional Stock Exchange website, you first have to find where to view shares rates, then to give the name of the share you want to check.

Vague queries

In this situation, a customer knows vaguely about a product he would like to purchase. For example, he knows a general category where to search for it. Traditional search algorithms rely on the sole description of the item to find it and results are not very good; such a behaviour

is reproduced by our algorithm when the search formula receives a high weight, and user and general interests have their weights set to nil. But if the weights are correctly balanced (which we hope to be the case), the interest of other customers will be taken into account, and this will lead to better results, because the system knows of both the current user's vague description and the targets other customers have been interested in, in this category.

Undecided customers

The algorithm may also help a user to find a starting point in his browsing. When a user logs in for the first time, he has no profile, but the nodes are reordered according to other customers interests. If a product or a category is particularly in the vogue, it will undoubtedly be "highlighted" by the system, thus giving some hints about where to start.

Knowledge advances by steps, and not by leaps.
Thomas B. Macaulay - Essays and Biographies. History.

3.1 Overall structure

The system is mainly an association of three tools commonly used in e-commerce today (Fig. 3.1).

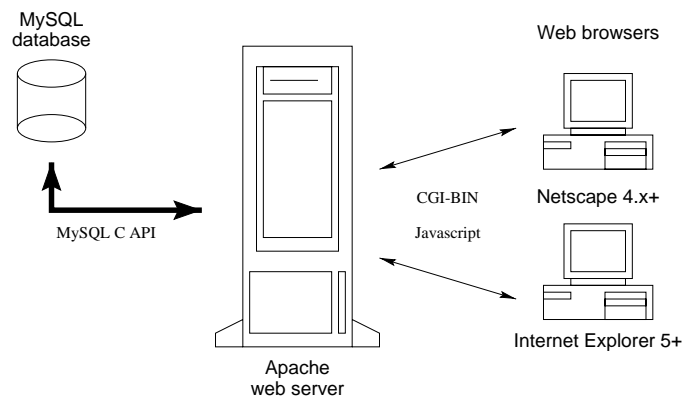


Figure 3.1: Overall structure

MySQL and C have been chosen due to their performance, and Apache for its modularity and stability on heavy network loads. Netscape Navigator 4.x+ and Internet Explorer 5+ are the most widely-spread browsers; moreover, special features like event management were required in the Javascript part, which earlier browsers do not provide.

3.2 Server side

The main work is to dynamically produce web pages, generated accordingly to the user's behaviour and to what is known about him and the targets. To achieve our goal, we need different informations to be stored in the database:

- We need to identify the users; the system provides a login mechanism. All the users are stored in a table containing their name, a login name, a password and a customer id;
- we need to store the semi-lattice structure and some information about nodes and targets, especially names and textual descriptions;
- we need to be able to log the user's behaviour within the structure so we can associate interest scores to the categories he's browsing;
- we need to analyse and synthetize raw logs to produce root-to-target paths, in order to spread scores over the structure.

Interaction between the server and the clients is realized through cgi-bin scripts written in C; A typical request to the server transmits a certain amount of information about the user's behaviour; this information is stored, processed using the algorithms introduced in chapter 2 and a complete HTML page is produced accordingly, all lasting less than a second, even with huge lattices containing hundred of thousands of targets ¹.

3.3 Client side

What seemed to be difficult, or even impossible to do at first sight, that is, monitoring very precisely user navigation through the website, turned out to be quite easy to implement, thanks to the great features provided by recent navigators, especially through Javascript and event management ².

There are two main problems with monitoring:

- How to establish a connection between the client and the server to transmit data, without using Java (Java engines on navigators usually have rather poor performance);
- Filtering the data and detecting moves such as a page reload or clicks on the "back" button.

It is not possible to establish network connections with Javascript. The only way to transmit data is to use cgi-bin scripts with parameters. On the other hand, cgi-scripts offer no interactivity and they are run on the server side. We need something having the same advantages as cgi-scripts and Javascript, but without the drawbacks...

In our implementation, we have managed to bypass the limitations of both cgi-scripts and Javascript by spicing up a bit the recipe with some DHTML. DHTML makes the link between the two, thus gathering together the interesting parts of each of them (Fig. 3.2).

¹speed is very important, our goal being to help a user to find a target quicker than with traditional browsing methods! Techniques developed to improve speed are detailed in [3], chap. 3.

²Big Brother matters are no concern of ours...

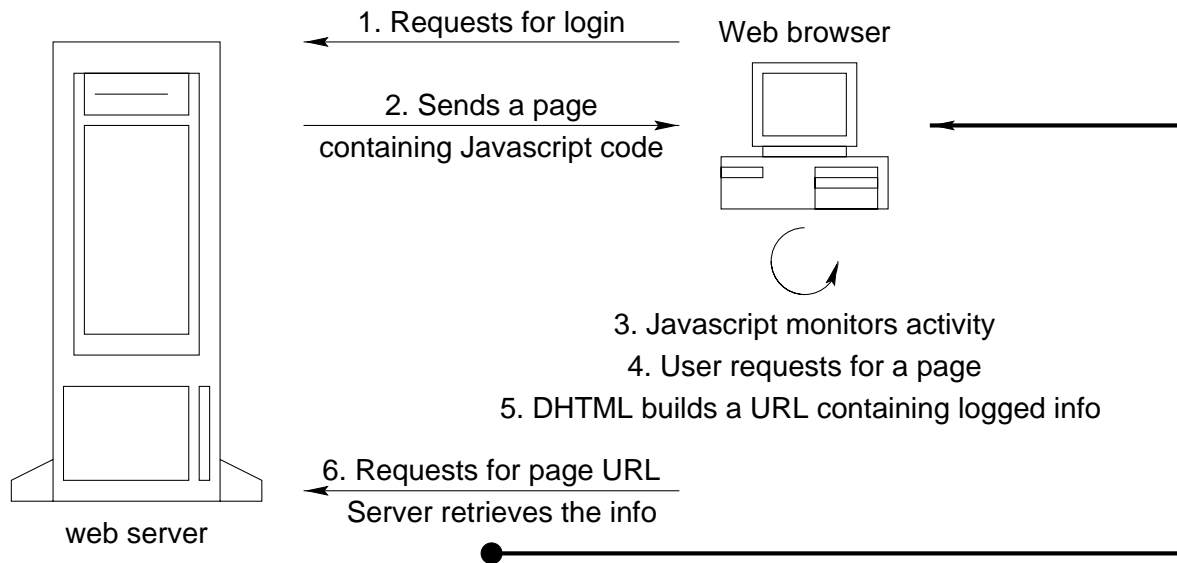


Figure 3.2: Tracking mechanism

To solve the problem with the user doing nasty things (such as reloading a page or clicking the “back” button, which prevents data transmission), the website is designed in such a way it recreates an environment similar to the one provided by the navigator as default, in a standalone window. Usual toolbars are hidden and replaced by the website’s, which are controlled by Javascript.

Once the user has logged out (if he just leaves his browser without logging out, a time-out mechanism allows for automatic logout), the server updates his profile by extracting interest scores from the raw interaction log and updating the mean scores on targets accordingly.

Genius is one percent inspiration and ninety-nine percent perspiration.
Thomas A. Edison - Newspaper Interview. Life (1932), ch. 24.

Main results

The main results achieved in matter of software engineering are the following:

- Using only cgi-scripts, Javascript and DHTML, it has been proven that it is possible to create an almost totally controlled browsing environment. The control script is absolutely transparent to the user and is able to dump large amounts of information, thanks to the powerful mechanisms which are now available in state-of-the art web browsers;
- Techniques have been developped to optimize access to databases via SQL queries, particularly when dealing with tree and lattice management;
- An architecture has been proposed so that the extended IOI concepts can be applied to databases containing up to hundreds of thousands of targets.

Robustness

The implementation we used for experimentations turned out to be much more robust than expected; It had been indeed feared that multiple users accessing the site at the same time might fool the server and that unexpected user behaviours could cause harm to log analysis.

The only severe problem which occurred was linked to the size of the buffers used to compose SQL queries. The amount of memory needed to build the queries was considerably under-estimated.

Efficiency

We have strong reasons to think that the modified IOI algorithm can help a lot when browsing very deep classifications. One more experimentation is yet to be made to gather statistics

about its real efficiency.

The important problem that has not received clear solution yet is how to adjust the multiple parameters (weights, thresholds...) of the algorithm in order to optimize its overall behaviour. In particular, it is not clear whether this should be left to the users so they can build their own profiles or whether it can be handled by the server alone.

One must also keep in mind that the system is strongly based on feedback: browsing trends are consolidated when the user gives a good mark to a target he has found thanks to the system. Entire regions of a hierarchy might be hidden to a user, if his habits are taken too strongly into account.

We have also noticed that the system, as it is implemented now, is extremely sensible to the categorization and the description of nodes and targets¹. The product database must be designed very carefully. An ill-designed database seems to involve poor results, probably for two reasons:

- Poor textual descriptions lead to confusion when looking for a target, so the algorithm will not be able to go down deep in the hierarchy;
- bad categorization prevents the algorithm to “find” regions with high scores because the distribution of correlated targets is too wide.

¹far more than expected.

CHAPTER 5

Conclusion

My lute, awake! perform the last
Labour that thou and I shall waste,
The end that I have now begun;
For when this song is sung and past,
My lute, be still, for I have done.
Sir Thomas Wyatt - To His Lute.

Novel concepts based on artificial intelligence is something considered as the next important step by companies in the field of e-commerce. Shopping web sites are still too static and too complex to browse.

During our work, we discovered that what we thought to be limited to applying a modified version of a well-known algorithm to product and service hierachies, was in fact extending to a much larger research domain. Many problems are still open and even unexplored; for example, how is it possible to optimize the classification of hundreds of thousands of products?

Tuning the system for better performance was also one of our main subjects of interest. Tuning turned out to be one of the most exciting part of the project, since we had no previous experience about it and had first to experiment with many configurations to be able to infer the system behaviour.

This work gave us a lot of different and new knowledges in the field of propagation algorithms, latticed structures, databases and web interaction, for finally allowing the realization of a demonstrator showing clear potential with the techniques implemented.

Database architecture

The first validation tests have been made on a fake product database containing about fifty targets. When validation was complete, this database was replaced by a bigger one (about one thousand targets).

The small database contains all the particularities that are usually found in latticed structures and complex trees (see chap. 2, sect. 2.3.4):

- Variable branch factor;
- Variable branch deepness;
- And most of all, multiple parents.

Raw data

The data is initially provided in a simple table, which is converted to the structure we have developed for storing the lattice by a small utility written in C.

Branch deepnesses higher than 1 are determined from the empty columns of the table (in fact containing the hyphen ("-") symbol). Multiple parents are determined by reproducing more than once a line with the same first columns in the table.

Textual descriptions for both the targets and the nodes are provided in additionnal tables, each containing two columns: the name of the target or node and a set of keywords.

Section1	Section2	Section3	Marque	Produit
Base elements	Cases	ATX towers	In Win	Midi tower ATX 230 W High quality
Base elements	Cases	ATX towers	In Win	Full tower ATX 230 W High quality
Base elements	Cases	AT towers	TourTek	Medium tower 200 W
Base elements	Cases	Desktop	DeskTek	Desktop ATX 230W High quality
Base elements	External	Monitors	Viewpoint	15p 0.28 mm 70 KHz
Base elements	External	Monitors	Viewpoint	17p 0.28 mm 70 KHz
Base elements	External	Monitors	Sony	19p G200 FD Trinitron 0.24 mm 96 KHz
Base elements	External	Monitors	Goldstar	15.1p TFT 500LC 61 KHz
Base elements	External	Keyboards	Mitsumi	SR W98 DIN / PS2
Base elements	External	Keyboards	Cherry	NTK SR PS2
Base elements	External	Mice	Logitech	Pilot 3 buttons serial
Base elements	External	Mice	Logitech	Pilot 3 buttons PS2
Base elements	External	Mice	Logitech	Mouseman II 3 buttons PS2
Base elements	Internal	Motherboards	Asus	P3B-F Intel BX 100 MHz 6x PCI 1x ISA ATX
Base elements	Internal	Motherboards	Asus	K7M AMD 751 Slot A 200 MHz
Base elements	Internal	Motherboards	Asus	P2B-DS Intel BX Dual Ultra2 Wide SCSI ATX
Peripherals	Mass storage	Hard disks	IBM	9.1 GB DMVS 4.9 ms 10'000 rpm 2 MB
Peripherals	Mass storage	Hard disks	IBM	37.5 GB Deskstar GP UDMA 66 5400 rpm
Peripherals	Mass storage	Hard disks	Quantum	9.1 GB Quantum Atlas IV 7200 rpm 2 MB
Peripherals	Mass storage	Backup	Iomega	Zip 100 MB internal IDE
Peripherals	Mass storage	Backup	Iomega	Jaz 2 GB external SCSI
Peripherals	Mass storage	CD-ROM drives	LiteOn	40x speed 6 MB/s
Peripherals	Mass storage	CD-ROM drives	Teac	32x speed 4.8 MB/s CAV
Peripherals	Multimedia	Graphic cards	Matrox	Millennium G400 32 MB
Peripherals	Multimedia	Graphic cards	ATI	Rage Fury 32 MB SDRAM TV-out
Peripherals	Multimedia	Graphic cards	Miro	Video DC30 Plus
Peripherals	Multimedia	Soundcards	Creative Labs	SoundBlaster Live 256 Value
Peripherals	Multimedia	Soundcards	Creative Labs	SoundBlaster Live Player 1024
Peripherals	Multimedia	Soundcards	Guillemot	Maxi Studio Isis
Peripherals	Multimedia	Soundcards	Turtle Beach	Daytona 32 PCI
Peripherals	Multimedia	Soundcards	Turtle Beach	Pinnacle A3D
Peripherals	Multimedia	CD-ROM drives	LiteOn	40x speed 6 MB/s
Peripherals	Multimedia	CD-ROM drives	Teac	32x speed 4.8 MB/s CAV
Peripherals	Network	Hubs	Soho	Basic Hub 8-port 10 Base-T
Peripherals	Network	Hubs	Kingston	Hub 5-port 10 MBps Base-T
Peripherals	Network	Modems	Zyxel	Omninet ISDN USB external 128 K
Peripherals	Network	Modems	3Com	Professional Message 56 K V90
Accessories	Disks	Zip	-	Cartouche Zip 100Mb
Accessories	Disks	Zip	-	Cartouche Zip 250Mb
Accessories	Disks	-	-	3"1/2 floppies
Accessories	Ink	-	Epson	Color ink Stylus 660
Accessories	Ink	-	Epson	Black ink Stylus 660
Accessories	Ink	-	HP	Color ink 690c
Accessories	Ink	-	HP	Black ink 690c
Accessories	Cables	Power	-	Extension 5"1/4
Accessories	Cables	Power	-	Extension 3"1/2
Accessories	Cables	Power	-	Y-cable 3"1/2
Accessories	Cables	Interfaces	-	Printer cable
Accessories	Cables	Interfaces	-	Serial cable

Figure A.1: Raw data from which the structure is built

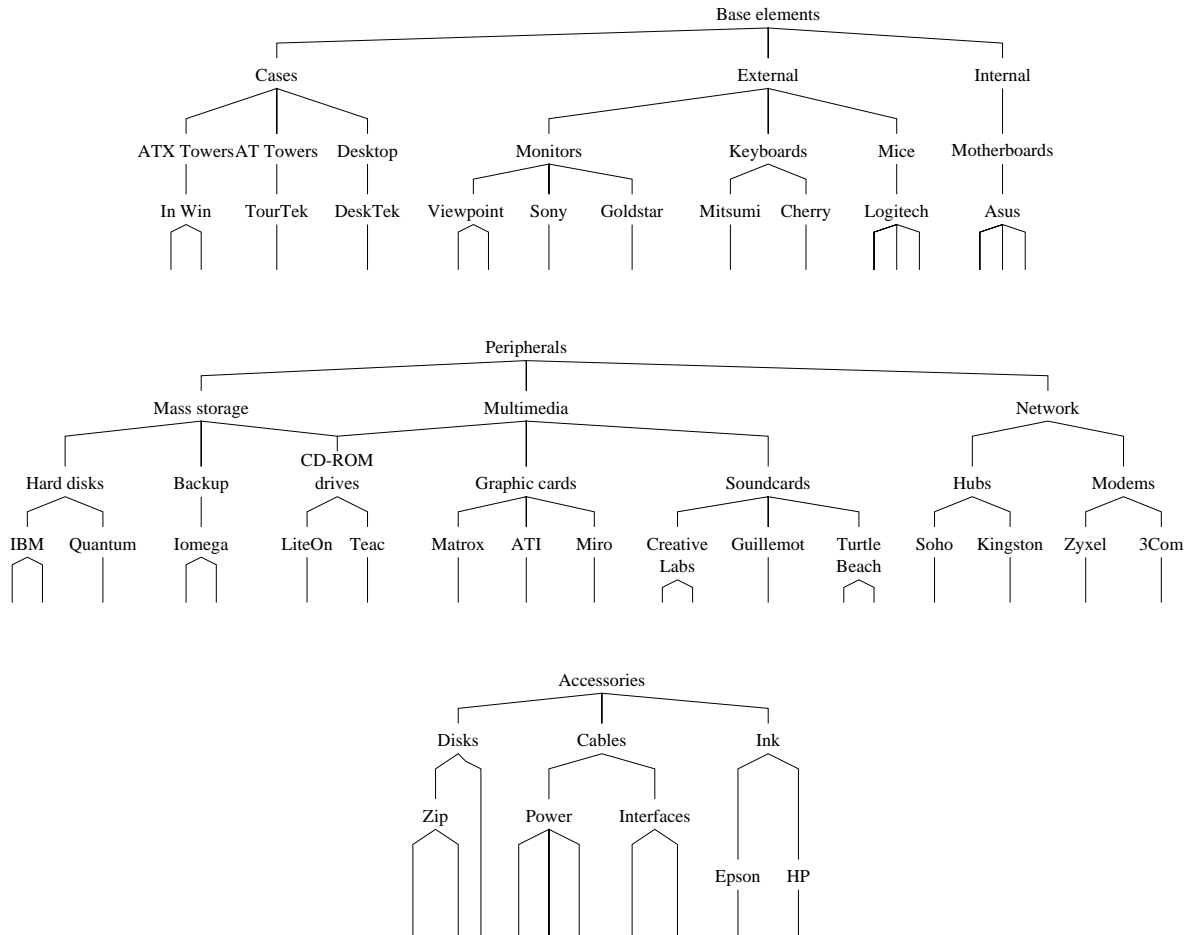


Figure A.2: The resulting classification

Bibliography

- [1] Uwe Gühl. *Design and implementation of a natural language based interface for the interaction with a command driven system - The Input/Output Interpreter IOI*. TELECOM Paris/RWTH Aachen. September 1994.
 - [2] Thomas Wolters. *Système d'aide par parcours d'arbre et recherche de mots clefs*. Mémoire de fin d'études.
 - [3] Edouard Forler. *Intelligent user interface for specialized web sites: an implementation*. Artificial Intelligence Laboratory, EPFL (internal report). March 2000.
-