Retrieval Based On User Behaviour

A.J. Kok
A.M. Botman

Department of Mathematics & Computer Science
Vrije Universiteit
De Boelelaan 1081
1081 HV Amsterdam, The Netherlands

ABSTRACT

This paper gives an overview of the ongoing research in the Active Data Bases project at the Vrije Universiteit, Amsterdam. In this project we are specifying and building a system that helps a user in his search for useful and interesting information in large, complex information systems. The system is able to do this, because it learns from the interaction about the users and the data it contains. The indications of the users are expressed in terms of interests in the data, which serve as building blocks for user and data models. These models are then used to improve the search for interesting data.

1. Introduction

There is a clear trend towards information systems of an increasing size and complexity. The associated problem is that it becomes almost impossible for a user to get a clear view of the structure and contents of the information in these systems, and henceforth to make the proper requests. It can be expected that these systems will be used more than current data base systems by non-professional or casual users which makes the problem even more difficult. These users will have even greater difficulty in exactly specifying what information they are looking for.

The problems encountered with current interaction methods are a result of the fact that the communication languages used are mainly designed to accommodate the machines, not the humans using it. The systems require exact and exhaustive specifications for retrieving information, while the user has often only a "vague" idea of what he is looking for. It has frequently been noted (e.g. [SIKL78, SIKL79, KAPL82, WEBB86]) that it is very difficult for a human to exactly describe what his interests are, how important every requirement is, and how and when to relax constraints. For instance, someone looking for a house to buy can probably not give a full and explicit description (query) of the house he would like. He would definitely be served with a system allowing him to give vague queries. In [SIKL79] an example is given about someone who wants to make a reservation by means of an airline trip system for a certain day. The system gives the requested information, but fails to point out that leaving a day earlier would make the trip 30% cheaper. The user just forgot to ask about the trips with an earlier departure date, while the system did not have the capability to realise

that the query of the user was more an indication of the preferred data than an exact request. Even queries that look exact, e.g. the number of employees of a department, might have vague aspects. For instance, the definition of employee could be nontrivial (part-time, temporary stationed, unpaid volunteer, etc.).

In human-human interactions it is self-evident that some constraints are more important than others, and that most people are willing to accept and even expect an answer which does not exactly meet the given requirements if it has some other, more favourable aspects (e.g. in the case of the airline trip a much lower price). The above mentioned examples clearly indicate that more active behaviour on the part of information systems is an essential requirement for them to be really useful. A system should realize that certain items which do not meet the constraints exactly might be more interesting than those that do, and should be shown also.

Another problem with current information systems is that they hardly use information about preferences that the user has given earlier. This information can be used to avoid repetitious interactions and to narrow down the amount of retrieved data in case of unspecific queries. For instance, if someone asks information about a rather general subject, e.g. about cars, then information about the kind of cars this person is usually interested in can aid in deciding what to show first. If no information about preferences of this user is known, then information about other users can be very helpful.

An information system can improve its interaction with the user by suggesting expected follow-up queries or even answering them. It is for instance often the case that someone who is looking for information about cars wants to know the addresses of the persons or garages where he can buy the most interesting ones. It is very helpful if the system suggests to retrieve that information or even displays those addresses immediately.

An improvement would also be the showing of the output in a more "intelligent" way. A user is seldom interested in long lists of data, so there is a definite need to summarize some of the results of an information request. An active system as we describe in this paper will typically try to show more information than a conventional system, so this need will be even greater. The form of such a summary should depend on the characteristics and preferences of the current user to make it as useful as possible.

The information density of the showed information increases also if it is made clear what the most interesting part of all that information is, for instance by forcing an ordering on it. If the information contains remarkable characteristics these should be shown in a way that will make them very likely to be noted by the user. The definition of what characteristics are remarkable depends of course again on the user.

All the problems mentioned above suggest that information retrieval systems would be much more useful and helpful if their processing was more controlled by the **interests of users in data**. Modelling of these interests would also match more closely the rather "vague" terms in which people seem to communicate. Therefore, we propose an interaction system that helps the user in three ways to obtain the most interesting information in an information system:

— It allows the user to specify his preferences in a simple way, namely by indications of interest and disinterest in information shown on the screen.

— Extra, related, information which is not explicitly requested, but which is probably useful, is shown.

— The retrieved information is shown as 'informative' as possible.

1.1. Activeness and impertinence

The system described in this paper, which is partially implemented, will help the user in obtaining the most interesting information. This is done by having the system build a model of its users by learning about their preferences. The system also builds a model of the information present in the database. These models give the system the opportunity to adapt itself to the current user, to let the system play a more active role in the dialogue, and to make it

possible for the system to be impertinent.

The **activeness** of the system as we see it, consists of giving extra information that might be more interesting than the information asked for, remembering preferences from previous dialogues with this user and with other users, and suggesting and answering follow-up queries that can be expected to be interesting to the current user.

The **impertinence** of the system consists of warning the user when new information has arrived at the data base that might be of interest to this user, even without a specific request for this. If for instance new information about second-hand cars is entered in the data base and the system notices that it contains a car that might be of interest to a certain user, it will notify him. This can be done by sending him mail, by telling him about it the next time he is using the system or by interrupting his current work, depending on the urgency. We have called this *imp*ertinent and *act*ive information retrieval system **IMPACT**.

1.2. User models extracted from the interaction

The models of the users and of the data are extracted from the interactions with the users. This means that regularities in the behaviour of people using this system are used to improve the interaction with it. The system learns about and from its users and thereby about the data it contains, and uses this knowledge to adapt itself to every individual user. No pre-programming of user models is necessary. We expect that for information systems to be accepted and used on a larger scale than now, by professionals as well as non-professionals, it is a prerequisite that they have an active and impertinent component. The systems should not be passive command-interpreters but play a much more important role in the dialogue.

The next three sections (section 2, 3, and 4) describe the models of the user, the interaction method, and the models of the data in more detail. Section 5 is a description of some important and necessary extensions to the currently implemented system. It gives an overview of what we are working on now.

2. Modelling the user

We think that certain patterns in the behaviour of people interacting with a database can be detected and used. These patterns are a result of inherent relations in the data as well as regularities in the personalities of people, resulting from social, political, or religious aspects. Our system does *not* extract or use the *reasons* for these patterns in people's behaviour, but does notice the *presence* of a regularity and tries to express this pattern in terms of interests. We want an active system to learn about preferences concerning the data to be able to predict interests of the user, which can then be used to retrieve information for him.

The interests of the user in the data form the basis of the model of that user as kept by our system. For instance, in the case of an information system with data about second-hand cars, the user model can contain the information that this user likes red cars, with a price around 5,000 Dutch guilders. If working with data about cooking recipes, the system will build user models containing preferences in ingredients, in cooking procedures and difficulty of the recipes. In a library system the models would include information about favourite authors, genres etc.

The user models need to be time dependent, because the users behaviour can change over time. These changes can have many reasons:

— A user usually has more than one subject to talk about to the system. At one time he inquires about second hand cars, a few moments later he could ask questions about insurances. The fact that he now wants to talk about insurances does not mean that he will never be interested in cars anymore, just that he is *now* more interested in insurances than in cars. It seems that a user has a current *focus* of interest and sticks to that for some time, i.e. hardly allows 'intrusion' of other subjects.

— The user may be looking for more than one optimal piece of information about a certain

subject for different reasons. For instance, he may want to retrieve information about cars for private use and for his company. Or he is looking for a car for his spouse and one for himself.

— The user may find some information during his search which he didn't know existed or which he didn't think of. Due to the active behaviour of the system information might be shown which is more interesting than the information he actually asked for, which will certainly influence his behaviour.

— The user's idea of some of the information in the data base might not agree with the actual information. For instance, initially he might be interested in a certain type of car, but after having seen some examples he disappointedly realizes that it is not exactly what he had in mind.

— The user formulates his ideas wrong. He could be thinking about the right notions but using the wrong "words". Other possible problems in this area are typing a wrong character or pressing a mouse button at the wrong place on the screen. The preferences of the user are not changed in these cases, but his behaviour is. This means our modeling strategy must allow for errors.

— External factors might influence his behaviour, for instance a pay raise could make him more interested in expensive cars.

— The user will have to get used to the system initially, resulting in different behaviour during a certain learning period.

The expected frequency of these changes in interests depends on the type of change, the user, and the domain. External changes for instance can be expected to occur seldom in the used cars domain: large pay rises or law changes favouring a certain type of car do not occur frequently. In the same domain, a car mechanic can be expected to know almost all there is to know about cars whereas a non-professional user might not know about certain very interesting types of cars. The interests of the latter person would consequently change more often, caused by unexpected information found in the data base. To be able to deduce statements about both the data and the user models, we assume that the chance of a change over small periods of time (in the order of a few interactions in a dialogue) is low and that the interests of a user do not change within one interaction.

The fact that a user's interests can change introduces a trade-off in the modelling of a user between how hard the system should stick to the subject and how fast it should react to changing interests. Most user modelling systems [RICH79a, CARB79, WAHL83] assume that the user's characteristics are more or less constant and that the models need only be refined, not really changed, during the dialogue. These systems usually employ large stereotypes which cannot be changed at all or only very slowly. Because they represent an average user of a certain type, the influence of the behaviour of one person will be low. People are almost always classified as belonging to one type, the need to combine information from two or more stereotypes seldom occurs.

One of the first prototypes of the IMPACT system [BOTM87] used a similar approach. The initial behaviour of people was compared with the information in a set of large stereotypes and, in case of a reasonable match, one was chosen to represent the user. When unexpected indications were given the system adapted the stereotype or, in case of very unexpected indications, a new stereotype was sought or created. It turned out that everything worked fine, as long as the user wanted to stick to the current subject. A change of interests of the user however lead to annoying and often unpredictable behaviour of the system.

From these results it was clear that a much more dynamic user model is needed. There are patterns in the behaviour of people, the reasons are mentioned in the beginning of this section, but they are much smaller than those represented by the mentioned stereotypes. A change of context, or a change of interests of the user should lead to the collection of information specifically suited to the new situation, instead of slowly changing the representation

of the old situation.

Therefore, in the current version of the IMPACT system we store the information about the user's interests in small, context dependent objects called **profiles**. When needed, the system tries to determine what the current context is (or in case of undeterminable or multiple contexts, a set of contexts) and collects information from profiles which agree with this context. The collection of all these pieces of information is accumulated in one large, profile-like object called the **focus**. This focus is the systems best model of the *current* interests of the user.

2.1. Structure of interest

The focus contains a set of statements about the user's amount of interest in data. An example of such a statement is "This users interest in red cars is 0.9". This number is called the **interest grade**. The interest grade is in the interval [-1,1], where a grade of +1 means "highly interested", 0 means "indifferent" and -1 means "strongly dislikes".

These statements can be made about object types (e.g. interest in cars), objects (a specific car), attributes (a colour-blind person is generally not interested in the colour of a car), values (dislike of the colour red) or a combination of those elements (e.g. someone might be interested in black BMWs, but not in black cars in general or BMWs in general).

Every statement has a **certainty factor** attached to it. This is a number in the interval [0,1] that reflects how certain the system is about the statement. For instance, a certainty factor of 0.9 means that the system is very sure about the correctness of the statement. A certainty of 0.1 means that the system has an indication that the statement might be true, but is hardly sure of it.

Whenever the system can deduce some of these interests, for instance after an interaction, one or more profiles will be created which contain these statements. The profiles have, just like the interests, a certainty factor attached to them. This is a result of the fact that commands might have more than one interpretation and that some of these interpretations are more likely than others. The extraction of profiles from interactions will be described in more detail in section 3.1.

A typical profile could be something like this:

```
Profile329:
 creationtime = "08:35 07/12/87"
 username = kate
 last-successful-use = "10:35 14/01/88"
 interests = {
  AV: make = BMW   grade = 0.7 cf = 0.5
  AV: make = Mercedes   grade = 0.8 cf = 0.6
  AV: make = Fiat   grade = −0.6 cf = 0.8
  COMB: (make = Fiat, colour = black)
            grade = 0.4 cf = 0.7
  ATT: number-of-gears   grade = 0.1 cf = 0.4
  ATT: make    grade = 0.7  cf = 0.8
  TABLE: cars   grade = 0.8 cf = 0.8
 }
```

This profile contains the information that the current user is interested in BMWs and Mercedeses, but not in Fiats. However, black Fiats are a bit interesting. The number of gears of a car is not important at all, while the make (obviously) is. If this or another user gives at a certain time a positive indication on Mercedes then this profile can be used as a small piece of evidence that in that case BMWs are probably also interesting, while Fiats are not. Here, the interest in Mercedes serves as context.

2.2. Profile selection and focus creation

The focus is the central component of the system. Every time new information about the user is derived, for instance after an interaction, the focus is computed anew. The focus is computed by selecting a set of profiles, and combining the information that they contain. The selection of the profiles is guided by the established context. This context is expressed in the interests extracted from the last interaction.

The profiles which will be used to create the new focus are selected as follows:

1) The newly generated profiles extracted from the last interaction are taken.
2) Profiles which were used to create the former focus are used again if they are not incompatible with the profiles from the first step.
3) Other profiles from the profile space (the collection of all the profiles of all the users) are used if they are compatible with the profiles collected in the first step.

Two profiles are incompatible if they contain interest statements that are incompatible. For instance, two statements about the same piece of data both with a high certainty factor but with very different interest grades are incompatible. Two profiles are compatible if they are not incompatible, *and* they have at least one pair of compatible interests. A pair of interests is compatible, if they are interests in the same piece of data and their grades are not incompatible. Note that compatibility of profiles is a stronger constraint than non incompatibility, so that compatible and incompatible are not complementary. For instance, two completely unrelated profiles are not incompatible, but also not compatible!

The information contained in the collected profiles is combined into a focus. In the combining, contradictory evidence for a statement (e.g. information that a certain user is interested in some data and another piece of information that he is not interested in it) results in a statement with a low certainty factor. Different pieces of similar evidence for a statement results in a high certainty factor for that statement. The weight of each piece of evidence (a statement from a profile) depends on the certainty factor of the statement, the certainty factor of the profile it came from, the time this profile was last used successfully and how often it is used successfully, how old the profile is, the "owner" of the profile (the user whose interactions led to the creation of this profile) and other factors. The interest combining algorithms used in the prototype system are based on those used in the GRUNDY system [RICH79b], which were based on the MYCIN certainty factor approach [BUCH84].

The profile selection and interest combining algorithms have several important characteristics: Step one of the selection algorithm assures that the system starts with the last derived information. This will enable it to respond quickly to changes in the interests of the user. Step two gives all the information of previous interactions which is not explicitly refuted by the user. In this way the user can follow a line of thought and seldom has to restate things, resulting in a smoother dialogue.

Step three will introduce information which was used in similar cases with this or possibly with other users. Together with step one and two this assures that one can change the subject quickly, and that information extracted earlier in the dialogue or in other dialogues and potentially useful in the new context is retrieved as soon as a reference to that context is made. A subject change is often introduced by a contradictory interaction, for instance the user gives an item registered as uninteresting a positive mark. Step one will notice the new object, step two will delete all disagreeing information, and step three will find older profiles which contain possibly relevant information concerning the new subject. The active behaviour of the system will be partly due to this mechanism.

Because of the way interests are combined and because profiles which were used to build the current focus are dropped as soon as their information is refuted, the decision which context is referred to is postponed as long as necessary. When it is not clear to which of the possible contexts is referred, information from all these context is used but, in case of inconsistenties, with very low certainty factors. As soon as a better reference is made, the wrong contexts and

all their information is dropped immediately.

2.3. Use of the focus in the system

The focus, the systems model of the current interests of the user, is the central component in the IMPACT system. It is used to decide what information to retrieve, what to show of this information and how to order and present it on the screen. Furthermore, it is used to interpret the commands of the user and can be used for several other purposes, e.g. to increase the efficiency of information storage.

The interest values in the focus are used to decide which information to retrieve and show to the user. In theory we could compute the interest for every item in the database and show the ones with the highest interest value, but this is not practical because the computation of item interests is rather complex. Therefore we use a two step procedure: First a database query is generated based on the main predicted interest values giving a set of favourable items which we call the **background solution**. In the second step, this background solution is analyzed more thoroughly and reduced to a **foreground solution**, which is shown on the screen.

The user's commands and indications are used to compute new profiles, which are used in the selection and combining of profiles to create a focus. The system then decides, based on the information in this focus, which attributes are best as a starting point for the constraints of a SQL-query. Best candidates are the most interesting attributes, and/or those that have large discrepancies in the interests in their values. This SQL-query is then send to the database process. The number of retrieved items should be around some optimal background solution size. This number is a trade-off between the chance of missing an item that might be interesting, and the costs and time of processing of all the items in the background solution for the more accurate interest computations. If the number of retrieved items is not near enough to the optimal background size then constraints in the SQL-query are added or relaxed. The optimal number can then be reached after several iterations of this process.

This point of the program is the place where the actual conversion of rather "vague" user statements and ideas are translated to the exact specification required by current database systems. The translation of the most striking interests to SQL clauses is rather obvious. For instance, if a certain user is extremely interested in red BMWs then add a clause which states those requirements. Problems arise when there are a lot of moderately graded interests of the user present in the focus. Items with one or two of those characteristics will usually not be interesting enough to be actually shown on the display. However, if one of the items in the database has several of those slightly interesting characteristics, then that item will probably be interesting enough and should not be excluded from the background solution. By choosing the optimal background size large enough (several times the optimal foreground size) and generating a sufficiently complex query the chance of missing such an item will be low.

When the number of items in the background solution is good enough, a more exact interest calculation takes place. For every item in the background solution the expected interest of the user in it is computed. Based on these interest grades the decisions are made which items are going to be shown by the system, how much of every item (e.g. which attribute-values) will be shown, and what summaries of information are going to be made.

If the focus does not contain information about the interest in this item (e.g. this user or other users have not said anything about this item in the current context in the past) then other sources for the interest computation are the interests in the characteristics of the item and the correlation between interests in this item and other information. We assume that if the interests in the attributes and the attribute-values of an item are known then the interest in the item can be computed. Problems arise however when one or more of the attribute-values are compound, e.g. the ingredients of a recipe are a *set* of items, each with its own interest. In this case, set contraction functions are needed which compute the interest in the ingredients set for a recipe from the interests in each of the ingredients itself. These functions are described in more detail in section 5.3.

If the interest in a certain datum still cannot be computed, for instance the interest in a non-compound attribute-value is unknown, then similarities can be used. These similarities represent the correlations between interests expressed in this value and in other data. If the interests in the other data are known a good guess of the interest in this datum can be computed. Similarities are described in section 4.

The interests in the focus are also used to make decisions about what information to shown on the display and how to show it. Summaries of the data are made in accordance with the focus. Summaries or groups are useful to increase the information density, e.g. the user does not have to look over long lists of data. How the summary is going to be made depends on the interests of the user. More detailed information about groups can be found in section 5.2.

3. The interaction method

The central role of the user model induces an interaction method that reflects the structure of the information in this model. In this way, the extraction of the model from the interactions will be much easier and the user will have better control of what is going on in the system. The main method of interaction consists therefore of indications of interest and disinterest in information shown on the screen. In the current version of the system the user gets the information in the form of tables on the display. Every piece of information on the screen is made sensitive to the mouse connected to the display. The user can indicate his interest by clicking the mouse buttons, one button for positive interest, another one for negative. In this way the user can indicate his interest by clicking on the item as a whole, on attributes, tables, attribute-values or combinations of these. The combinations are indicated by first clicking a "start combination" button, then indicating the elements of the combination and then clicking the "end combination" button. When the user thinks he has said enough and wants the system to go on, he should click the "continue" button or after a waiting period the system decides to continue with the commands given up to now. An example of a screen is given on the next page.

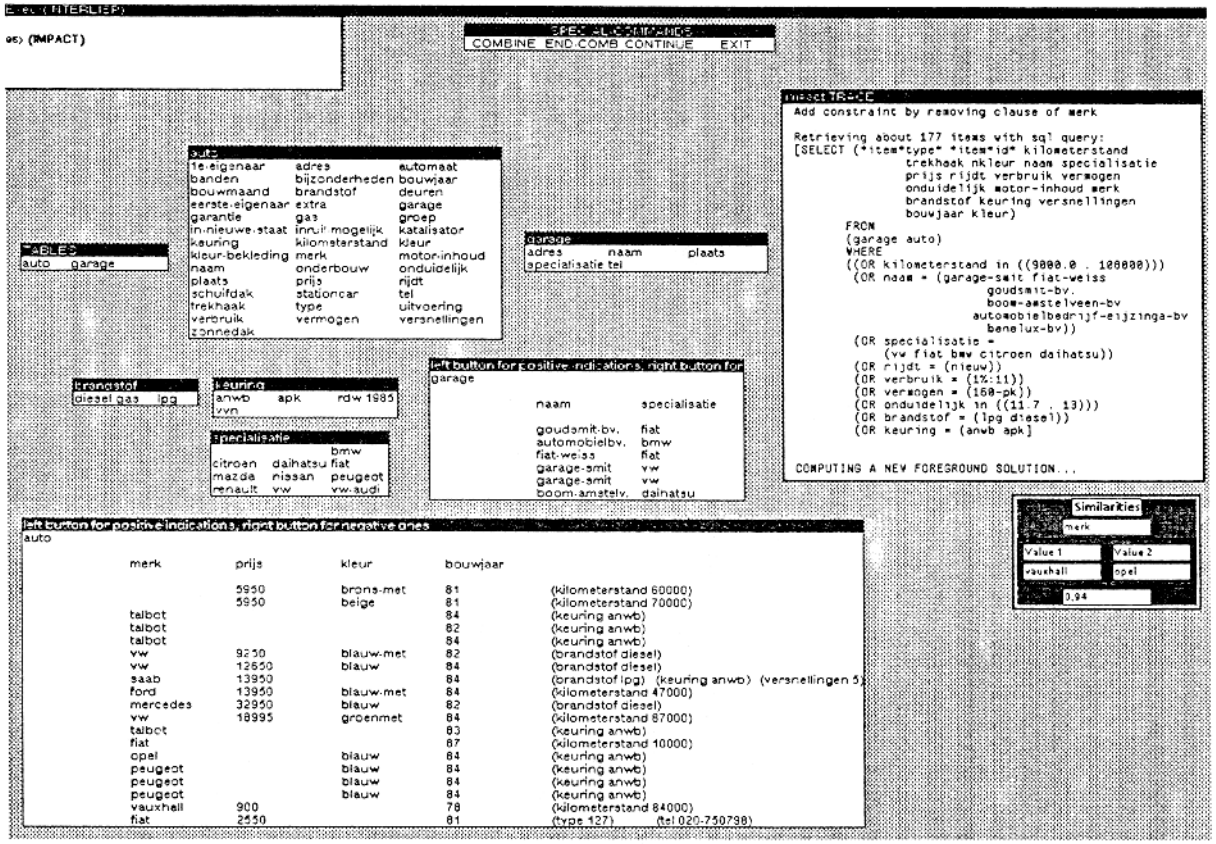
Another way for a user to express his preferences is by typing a commandline to a "normal" SQL-interpreter. This is a good vehicle for small and well defined queries, but for more complex and less strict ones (the "vague" queries) the method working with indications is much easier to use. Whatever method is used, in general the indications and commands are never executed directly but are used to adapt the system's model of the user. This model is then used to retrieve new information. This means that for both methods the system will behave "actively".

There seems to be a need to be able to give "exact" specifications besides the "normal" interest indications [LARS87]. This is for situations in which the user does know exactly what he is looking for. In the current version of the IMPACT system the preferred type of behaviour can be obtained by choosing the active or the exact mode. It is not yet possible to mix exact and active indications in one interaction.

3.1. Extraction of profiles from the interaction

The statements about interest are extracted in various ways from the indications. A simple first heuristic is that most indications can be converted directly to interests. If someone gives a positive click on "yellow" he is probably interested in that colour even though this is not 100% certain: he might have made a typing error or misunderstood the information on the screen. A second heuristic is that an indication in a datum probably means that the comprising data are interesting too. For instance, a positive indication of the colour yellow in the table of cars means that the attribute colour itself is probably interesting also, because the user based his choice on it. These interests have of course a lower certainty than the interests directly indicated (in this example the attribute-value yellow).

When multiple items are indicated, frequencies of the attributes and values of those items are computed to determine which of them were possibly used as selection criteria by the user. In


(IMPACT)
SPEC-ELIMINANDS
COMBINE END-COMB CONTINUE EXIT
auto
1e-eigenaar adres automaat
banden bijzonderheden bouwjaar
bouwmaand brandstof deuren
eerste-eigenaar extra garage
garantie gas groep
in-nieuwe-staat inruil mogelijk katalisator
keuring kilomsterstand kleur
kleur-bekleding merk motor-inhoud
naam onderbouw onduidelijk
plaats prijs rijdt
schuifdak stationcar tel
trekhaak type uitvoering
verbruik vermogen versnellingen
zonnedak
TABLES
auto garage
garage
adres naam plaats
specialisatie tel
IMPACT TRACE
Add constraint by removing clause of merk
Retrieving about 177 items with sql query:
[SELECT (*item*type* *item*id* kilometerstand
trekhaak nkleur naam specialisatie
prijs rijdt verbruik vermogen
onduidelijk motor-inhoud merk
brandstof keuring versnellingen
bouwjaar kleur)
FROM
(garage auto)
WHERE
((OR kilometerstand in ((9000.0 . 100000)))
(OR naam = (garage-smit fiat-weiss
goudsmit-bv.
boom-amstelveen-bv
automobielbedrijf-eijzinga-bv
benelux-bv))
(OR specialisatie =
(vw fiat bmw citroen daihatsu))
(OR rijdt = (nieuw))
(OR verbruik = (1%:11))
(OR vermogen = (150-pk))
(OR onduidelijk in ((11.7 . 13)))
(OR brandstof = (lpg diesel))
(OR keuring = (anwb apk]
COMPUTING A NEW FORESGROUND SOLUTION...
brandstof
diesel gas lpg
keuring
anwb apk rdw 1985
vvn
specialisatie
bmw
citroen daihatsu fiat
mazda nissan peugeot
renault vw vw-audi
left button for positive indications, right button for
garage
naam specialisatie
goudsmit-bv. fiat
automobielbv. bmw
fiat-weiss fiat
garage-smit vw
garage-smit vw
boom-amstelv. daihatsu
Similarities
merk
Value 1 Value 2
vauxhall opel
0.94
left button for positive indications, right button for negative ones
auto
merk prijs kleur bouwjaar
5950 brons-met 81 (kilometerstand 60000)
5950 beige 81 (kilometerstand 70000)
talbot 84 (keuring anwb)
talbot 82 (keuring anwb)
talbot 84 (keuring anwb)
vw 9250 blauw-met 82 (brandstof diesel)
vw 12650 blauw 84 (brandstof diesel)
saab 13950 84 (brandstof lpg) (keuring anwb) (versnellingen 5)
ford 13950 blauw-met 84 (kilometerstand 47000)
mercedes 32950 blauw 82 (brandstof diesel)
vw 18995 groenmet 84 (kilometerstand 87000)
talbot 83 (keuring anwb)
fiat 87 (kilometerstand 10000)
opel blauw 84 (keuring anwb)
peugeot blauw 84 (keuring anwb)
peugeot blauw 84 (keuring anwb)
peugeot blauw 84 (keuring anwb)
vauxhall 900 78 (kilometerstand 84000)
fiat 2550 81 (type 127) (tel 020-750798)

this algorithm the current interests of the user and the data model described in section 4 are also used.

At the moment we are working on profile abstraction methods. These methods can be used as interest extraction algorithms. The general idea is to translate the given indications directly to profiles, and to define methods which abstract new profiles from these profiles. The algorithms that now extract profiles from indications in an indirect way (like the frequency counting algorithm) will then be subsumed by these.

4. Data model - similarities

At various places in the system we need to know how "similar" two pieces of data are to each other. Examples are:

- To compute the interest of a datum based on the known interests of other data. If no information is available about the interest in this datum in the current context, then its interest can be computed from similar data of which the interests are known.
- To interpretate the user commands in the interest extraction algorithms. For instance, the system needs a metric on the attribute price to recognize the situation in which a user only chooses "cheap" cars.
- To determine whether a certain indication is "out of character".
- To generate summaries of data. The classification of values of an attribute should be based on how similar those values are to each other, so that for instance all the dark colours will be in one group.

The **similarity** of two pieces of data can be obtained by computing correlations in the indications of users. These correlations indicate the direction and strength of similarities in appraisal of people in these data. For example, when white and yellow cars are often graded the same (e.g. people indicate them both positively or both negatively) then white and yellow will obtain a high similarity value. Also, when people often approve of red and disapprove of pink cars then pink and red will get a negative similarity value. Similarities can in this way describe which colours look alike and which are different. It can also describe that prices of $990 and $995 are "closer" to each other than $1001 and $998. The system would observe different behaviour concerning those two pairs, but would never ponder about the psychological reasons for this.

The correlation in grading between two objects is a number between +1 and -1, with +1 meaning that the objects are always graded the same and -1 meaning they are always given opposite interest indications. A similarity of 0 means that there is no evidence for a correlation at all in the expressed or deduced interests in those two data.

At the moment, the similarities describe how the average user sees the data or, more precisely, what the consensus of the users about the resemblance of the data is. Personalized views of the data are not represented in the current similarities. However, personal differences in interest in the data are adequately represented in the profile space.

4.1. *Computation of similarities*

Since the similarities give a more general view of the data model and therefore can not be expected to change fast, it is not necessary to update this model after every interaction. Instead the rather complex algorithm is executed in idle time, for instance at night. For every possible pair of data, all the interest grades of every user in that pair are collected. This gives a list of pairs (interest in datum1, interest in datum2) together with a weight for every pair. This weight is an indication of how much this pair should contribute to the correlation computation.

The weight of the interest pair depends on several factors. For instance, how far apart in time the two data were valuated. Another factor is how sure the system is of the interests. If the

certainty factor of one or both of the interests is low, then it should be considered less convincing evidence than with two very certain interests. Several of these heuristics are used to compute the weight of the interest pair.

The difference in knowledge represented by profiles and by similarities should be clear by now. The profiles contain information about what the interest of the user in a certain datum was in a certain context, according to the system, while the similarities express a *tendency* in the valuations of two pieces of data.

A similarity can in principle be computed between any pair of objects. In the current version of the IMPACT system, only the similarity between attribute-value pairs on the same dimension is computed. This enormously reduces the complexity of the algorithm and the storage needed for the similarities, and seems to be enough for our current uses of them.

When IMPACT cannot find a similarity value it needs, it can resort to some default algorithms. These can depend on the type of the value. For instance, for the year of construction of a car, an integer value, the difference function can be used. The similarity between addresses can be based on the zipcode, etc. These defaults make the behaviour of the system in the first period of use, when it has not learned much yet, somewhat less erratic. Default similarity functions can be given by the database administrator, thereby taking a role somewhat similar to a knowledge engineer.

Similarities represent a very simple data model, in fact nothing more than a metric on the data. At the moment we seem to be able to do enough with only this kind of knowledge. We specifically do not want to abstract rules or derive other kinds of "deep" knowledge as we think that real problem solving is a task for the underlying information system, and IMPACT is only the interface to it. The IMPACT system might however be helpful in choosing the right problem solving method, in searching for the relevant attributes, etc.

5. Current work

This section describes some of the issues we are working on at the moment.

5.1. Computed attributes

Information is not always stored as pure data. To get a specific piece of information from an information base, systems sometimes have to perform transformations, calculations or use rules to derive the answer. In this view, expert systems, knowledge bases, as well as ordinary databases are all information systems. To be able to handle these kinds of information IMPACT uses **computed attributes**. The computation of such an attribute can, depending on its type, be executed within IMPACT, the underlying database system or an external agent. A "normal" attribute can be viewed as a degenerate case of a computed attribute, viz. a simple computation (lookup) executed in the underlying database. Other examples of computed attributes are: showing data in other units (miles versus kilometers, degrees versus radians), computing prices in other currencies using stored exchange rates, averaging or summing of values (for instance, the average age of women in each department). More complex computations could be invoking numerical algorithms or expert systems, accessing distant information bases, etc. These functions can be brought in beforehand by the database administrator or they can be generated by the IMPACT system itself, using these predefined functions and the interests as building blocks. For example, if a user seems to be interested in the age of people in a certain group, then it could suggest to compute the average age of them, etc.

Every computed attribute will have a cost factor associated with it, which is an indication of the amount of time, space and/or money it will take to compute this value. When the value of an attribute should probably be shown but the system is not sure it is interesting enough to compute its value considering the costs, the system will show a special token in its place. If the user gives a positive indication on this (not yet computed) value, it will become more interesting and will consequently be computed. If on the other hand the value is ignored or

given a negative indication then it will not be computed and will disappear eventually.

5.2. Groups

The classification of values into **groups** has several advantages. It can increase the information density of the shown information, it enables the user to give more global indications so that larger "steps" in his search can be taken, and it enables the user to give indications about values that are not present in the shown list of items.

In [WIED84] a good example of fruitful use of summarizing is given. A question is posed to a system with information about ship movements:

Which ships travelled from Shanghai to Hong Kong last week?

Since the result turned out to contain 318 vessels, a more informative response was given by classifying these:

318 ships
---
1 freighter: IMANU MARU
317 fishing boats

IMPACT will be able to make these summaries based on the user and data models. The most interesting attribute is the best candidate to make a summary on. The actual classification of the value set into groups is also based on interests and similarities, e.g. the most interesting values are put in separate groups, not so interesting and similar values are put in the same group, etc. The decision what to show of the elements is again guided by the user's interests. Since the system recognizes that the fishing boats are not interesting at all, nothing is shown about them except for their type, in contrast to the interesting freighter. Finally, the order in which the information is shown is also based on the user model.

From this example not only the importance of groups is clear but also the strength of the use of a model of the user's interests. This model is used here for four decisions: on which attribute to base the summary, how to make that summary, what to show of the items in the different groups and in what order to show the groups.

5.3. Contraction functions

How do we predict the amount of interest a person has for a particular item in the database? If the user has expressed his interest for this item in the past, we can use that as a starting value. But in any nontrivial database, we must assume that the user has not seen or indicated most of the items. We would like to have a method to predict in general what the interest in an item is.

We assume that a good prediction of the interest in an item can be computed from the interests in its characteristics. E.g. the interest in a car can be computed from the interest in its colour, make, age, price etc. It is obvious that not every attribute has the same influence, and that the relative importance of attributes may vary between persons. In the current version of the IMPACT system we compute the interest in an item by taking the sum of the interests in the value of each attribute weighted by the importance of that attribute. As importance of an attribute we take its interest value, under the assumption that a person is interested in the attributes he thinks are important and vice versa.

An alternative to splitting the interest grades over attributes and values is to use *only* the interest grades of values. However, this has the disadvantage that when a person changes the subject the system would have to change interest values for every possible car colour. In our scheme we only have to change the interest in the attribute. It is also closer to our opinion that the interest distribution is context dependent but that his interest in the various colours of a car remains the same, even though this part of his interest distribution is not active at the

moment. Therefore it is better to disable the context, by decreasing the interest in the attribute colour itself, instead of the interests in the colour values.

With this model of interest we can describe and predict to a fair extent the interest in simple items. But it is not rich enough to describe the interest in compound objects, objects with a *set* of values for a certain attribute. Take for example the amount of interest a person has in a compact disc. Clearly some attributes which are influential are its cover-picture, price, state of the box etc., but the most important components of a disc are the songs on it. A problem is how to compute the interest in a set of songs. Is a set of 2 good songs and 10 bad songs less or more interesting than a set of 12 mediocre songs? Another example is the interests in cooking recipes, which obviously depend on the interests in the various ingredients. A person could like a recipe if it contains only a few tasty ingredients, or he could require that he likes all the ingredients. Another person would be satisfied when a dish contains no ingredients that he strongly dislikes. People on a low-salt or low-fat diet have again other constraints.

We try to model the interest in compound objects by defining **set contraction functions.** These are functions which compute the interest of a set of items from the interest values of the elements of that set. Some example functions are

— The average interest of the elements.

— The maximum interest of the elements.

— The minimum interest of the elements.

— The average of the highest 25%.

— The average of the highest 30% minus the average of the lowest 30%

..

We are investigating two possible ways of dynamically obtaining these functions: the "background" and "foreground" approach (this has nothing to do with the background and foreground solution!). In the background approach we see these functions as belonging to the data model. For every compound attribute there is a function which describes the interest relation for a particular user between the "main" structure and the components. Analogous to the computation of similarities, there is an algorithm running in the background which tries to find a pattern in the interest values of the major and compound structures. We think that a good algorithm for this is one using "genetic programming" strategies [GREF85], requiring functions like the ones above as building blocks. The algorithm continuously creates small mutations of these functions and tries them out. A drawback of this approach is that it needs several test cases to reach some conclusions, which means that it takes a while for the system to notice a pattern.

The foreground approach tries to find the contraction function during the dialogue with the user. It consists of adding beforehand for each compound attribute a set of new computed attributes each of which computes the interest in the set defined by the compound attribute. In principle, the indication of the user of such a computed attribute leads to the use of the corresponding contraction function. However, it is not necessary that the user himself chooses it because the proper computed attribute can be chosen by the system without even showing it to the user. This is because the system, when processing the user commands, can check also the non-visible but possibly important attributes for correlation with the users indications of interest. If such a correlation is found, the attribute will gain importance automatically. This has the advantage of giving the user a way of checking and possibly more explicitly influencing the system in the refinement of his model. This method also has the advantage that it will probably find a contraction function faster than the background method, but at the cost of greater overhead during the dialogue. An advantage of the background method is that it will probably find a better solution on the long run as it will always try to improve on the functions and uses more data to check them on.

5.4. Impertinence

The current version of the IMPACT system assumes that the data base does not change. We are currently working on extending this to a data base in which new data is entered and data is deleted. One of the consequence of the entrance of new data is that the system should be **impertinent**. It should check the new data and related data in the data base to see if some of it has become interesting enough for a certain user to warn him about it. The information in the focus is in itself not enough to recognize such a situation, because the focus contains only information about *current* interests of the user. If the user is now talking about cars, then all the information about his interests in houses will be absent from the focus. If however new data about a house that exactly meets his requirements enters the information system, then this user would probably want to be told about this. This indicates that some profiles or groups of profiles remain important even when they are not used to create the current focus. New data should be compared with the information in these "important" profiles, and if the match is successful and the profiles important enough then the user should be warned.

6. Conclusions

The interaction with information systems in large and complex domains needs improvement. This can be obtained by letting the system play a more active role in the dialogue with its users. For this, the system needs models of its users and of the data it contains. The results in working with the prototype version of IMPACT indicate that adding an active component to an information system can be really helpful in retrieving the most interesting information.

Acknowledgements

Professor Laurent Siklóssy supervises this project. It is partially supported by the Netherlands Organization for Scientific Research (NWO) under grant 612-319-008 to Prof. Siklóssy.

References

[BOTM87]. Botman, A.M., Kok, A.J., and Siklóssy, L., "Using user models and automatic expertise acquisition in man-machine interaction," pp. 389-398 in *Proc. Expert Systems and their Applications*, Avignon (1987).

[BUCH84]. Buchanan, B.G. and Shortliffe, E.H., *Rule-based expert systems: The MYCIN experiments of the Stanford Heuristic Programming Project*, Addison-Wesley Company, Reading, Massachusets (1984).

[CARB79]. Carbonell, J.G., "Subjective understanding: Computer models of belief systems," (Research report #150), PhD thesis, Yale University (1979).

[GREF85]. Grefenstette, J.J.(ed.), *Proc. of an Int. Conf. on Genetic Algorithms and their Applications*, CMU, Pittsburgh, PA (1985).

[KAPL82]. Kaplan, S.J., "Cooperative responses from a portable natural language query system," *Artificial Intelligence* **19** , pp. 165-187 (1982).

[LARS87]. Larsen, H.L., "Knowledge representation in IRIS, an information retrieval intermediary system," pp. 529-548 in *Proc. Expert Systems and their Applications*, Avignon (1987).

[RICH79a]. Rich, E., "Building and exploiting user models," pp. 720-722 in *Proc. of the Int. Joint Conf. On Artificial Intelligence (IJCAI)* (1979).

[RICH79b]. Rich, E., "Building and exploiting user models," (CMU-CS-79-119), PhD thesis, Carnegie-Mellon University (1979).

[SIKL78]. Siklóssy, L., "Impertinent question-answering systems: justification and theory," pp. 39-44 in *Proc. of the 1978 ACM Annual Conference*, Washington DC (1978).

[SIKL79]. Siklóssy, L., "Passive vs. active question-answering," pp. 271-276 in *Proc. of the First Int. Symp. on Policy Analysis and Information Systems*, Durham, N.C. (1979).

[WAHL83]. Wahlster, W., "Overanswering Yes-No questions: extended repsonses in a NL interface to a vision systems," in *Proc. of the Int. Joint Conf. On Artificial Intelligence (IJCAI)* (1983).

[WEBB86]. Webber, B.L., "Questions, answers and responses: interacting with knowledge-base systems," pp. 365-402 in *On knowledge base management systems*, ed. J. Mylopoulos (1986).

[WIED84]. Wiederhold, Gio, "Knowledge and database management," pp. 63-73 in *IEEE Software* (Jan. 1984).