Adaptive Profile Inferences for Cooperative Query Answering

Michael Barg and Raymond K. Wong
Basser Department of Computer Science
The University of Sydney
NSW 2006, Australia

Abstract

In response to the differing relaxation needs of individual users, this paper presents novel solutions for generating improved user profiles. A method of merging stereotype information and individual profiles is suggested, so that the system can react usefully upon first encountering an individual. This paper also proposes a method whereby the system can make inferences about useful profiles to use in a given situation, in instances where the exact situation has not been previously encountered, but where similar situations have. Finally, a method is proposed to re-evaluate the inference strategy of the system in determining the importance of the various terms for a specific user, instead of using only a single, rigid inflexible rule.

1 Introduction

Databases and knowledge base systems are often difficult to use because they do not attempt to cooperate with their users. A database or a knowledge base query system provides literal answers to queries posed to them. Such answers to queries may not always be the best answers. Instead, an answer with extra or alternative information may be more useful and less misleading to a user. To achieve that, a query answering process should collaborate with users to find the information that they are seeking. When users ask queries that do not obtain the information that they seek, a technique called relaxation [7] was proposed to enable a knowledge-based system to work interactively with a user to find alternative answers that are related to the answers of the original query.

For instance, the Cobase group [4; 5] has explored a method for generalizing queries in order to provide related answers. Generalization relies on having explicit hierarchies of predicates and terms over the language. A hierarchy is represented as a partial order. A query rewrite is accomplished by replacing predicates and terms from the query with corresponding predicates and terms higher in the hierarchy. The resulting query is considered to be more general. Such a structure is called the type abstraction hierarchy [3] in Cobase. The hierarchy covers three ways to do ordering: subsumption, composition, and abstraction.

For a description of the state-of-the-art of cooperative answering systems and other work on relaxation in cooperative answering systems, read [6; 11] and [7], respectively.

Although relaxation provides relaxed answers to users, it is a general approach to seek additional answers to a query that may or may not be of direct interest of the user. To ensure that the additional answers are relevant, some schemes posed by others for detecting users' plans and intents may be used to refine the approach. These include the works on analysing intention [1], user modelling for responding to misconceptions [10], user modelling for guiding relaxation [2] and generating cooperative answers with respect to users' goals [15]. In the system FLEX, Motro proposes allowing the user to select directions of relaxation, and thus to indicate which relaxed answers may be of interest [14]. Cuppens and Demolombe suggested the way to identify topics of interest which is compatible with the relaxation approach. Finally Minker and his students developed a relaxation mechanism based on taxonomy clauses in a deductive database [8; 9].

Previous work has recognised the fact that differing individuals will have different relaxation needs. A relaxation which is acceptable to one person may be completely unacceptable to another individual who asked exactly the same query.

Some models [5; 12] deal with this by explicitly asking the user to add all of their preference criteria to the query itself. This has a number of disadvantages. Firstly, this is very tedious for the user to do every time (increasing the chance that they just will not bother). Secondly, it assumes a certain level of expertise in formulating formal database queries. Whilst this may be of no problem to the computer scientist, it poses serious problems to peo-
people who are much less computer literate and yet who still frequently use databases (now more than ever with the advent of the web, where many users of differing computer expertise send database queries all the time). Those without the required level of expertise (which is the majority of Web users) would be unable to benefit from the full effectiveness of such relaxation.

Gaasterland [7] and Motro[13] use the notion of Constraints, (relaxations that the user considers unacceptable, or other criteria they do not wish relaxed at all) which are appended internally to the query. However, they seem unclear on how precisely these constraints are to be obtained by the system. It is implied that these constraints be explicitly given to the system, in which case the objections as stated above apply. Furthermore, whilst this does restrict the unacceptable relaxations, it in no way helps the system deal with the acceptable relaxations (i.e., which should be relaxed first? Which relaxation is more acceptable to the user?)

Additionally, Gaasterland [7] has devised a system which determines relaxation order based on misconceptions about the database which the user may have had and which are reflected in the query. His methodology, however, does not take into accounts situations in which the user has no misconceptions but still needs their query relaxed, which is a quite common occurrence.

Barg and Wong [2] have suggested building a user profile based on observations of the individuals query behaviour, and using this profile to guide relaxations. The system uses the notion of importance to determine the order in which terms within a query should be relaxed. The notion of criteria preference is used to determine which relaxations are acceptable, and within these acceptable relaxations, which are preferable.

This paper addresses some deficiencies in the above mentioned user modelling process. It proposes to make use of stereotype information when a user initially encounters the system, so as to provide some guidance for initial queries, and to diminish reliance upon these stereotypes as the system learns more about the specific user. The paper also proposes a method whereby the system can make inferences about appropriate relaxation preferences in situations for which the user issues a query containing a combination of terms the system has not encountered before, but where there exist within the system examples from the users history dealing with queries which are in some sense "similar". Finally, the paper presents a system of continuously evaluating the inference rules it uses in order to determine "importance" of terms for a particular user. The system learns and responds to the specific users mode of indicating importance, rather than relying on a single, static, inflexible rule.

The organization of this paper is as follows. In section 2 the authors discuss the Logical Abstraction Hierarchy, a representation used to enable the system to determine which terms are logically similar to each other. Section 3 proposes a method of inferring an appropriate profile to use in a particular instance where the system has not had any direct experience of this instance before, but has had experience of instances which are in some sense "similar". In section 4, the paper presents a method for re-evaluating the inference rules that the system uses to determine the importance of terms for a user. It learns and adapts by looking at the users own behaviour, rather than relying on a static, inflexible rule. Finally, in section 5, the ideas presented in this paper are summarised and an overview of the work given.

2 Logical Abstraction Hierarchy

A Logical Abstraction Hierarchy (LAH) is a modified extension of the Type Abstraction Hierarchy (TAH) [5]. The LAH denotes a logical hierarchy, comprised of abstract classes, attributes and categorical values for attributes. Such entities are arranged in a tree structure to represent which entities are logically superordinate and subordinate to the others. Figure 1 shows an example of an LAH. Note that some nodes in the LAH are attributes (vehicle, restaurant, price), whilst others are categorical values that certain attributes can assume (car, boat, motorcycle, 4-cylinder, 6-cylinder). Furthermore, some are abstract notions (transport), not directly represented in the database itself. Note that numerical attributes (price) are never subdivided further.

The main function of an LAH is to allow determination of logical proximity, where the logical proximity of two nodes, a and b, is defined as the number of nodes contained in the path from a to b. By considering the logical proximity between a and b and between a and c, it is possible to determine an ordinal relationship between the logical relatedness of c and b to a.

For example, consider the node vehicle.boat in Figure 1. The logical proximity between vehicle.car.4-cylinder and vehicle.car.6-cylinder is 1. The logical proximity between vehicle.car.4-cylinder and vehicle.boat is 2. Thus
it is possible to state that vehicle.car.4-cylinder is more closely related to vehicle.car.6-cylinder than vehicle.boat. Note that it is only possible to make ordinal statements (more/less closely related to) using the LAH. The specific number which represents the logical proximity has no intrinsic meaning in itself, as 2 different LAHs may represent the same concept but with differing levels of detail.

When the notion of logical proximity is combined with an awareness of depth, then the system can start to make inferences regarding both closeness of relationship and the relationship between the two entities. For example, the logical proximity between vehicle.car.4-cylinder and vehicle.car.6-cylinder is 1. The logical proximity between vehicle.car.4-cylinder and vehicle is also 1. This does not mean to imply that both vehicle.car.4-cylinder and vehicle stand in the same relationship to each other, however. All it implies is that the relationship between siblings vehicle.car.4-cylinder and vehicle.car.6-cylinder is closer than any other siblings which would involve the connecting path travelling through vehicle (such as vehicle.boat.price).

The main use of the LAH is to see how closely related entities are which share the same name but occupy a different node in the LAH. The system will use the logical proximity (along with other tools as described later) to infer how information relating to one node is likely to be related to a different node with the same name. For example, consider vehicle.car.6-cylinder.price. The logical proximity between vehicle.car.6-cylinder.price and vehicle.car.4-cylinder.price is less than the logical proximity between vehicle.car.6-cylinder.price and vehicle.boat.price. Thus the system infers (in the absence of any further information) that the information contained in vehicle.car.4-cylinder.price is more likely to be relevant to vehicle.car.6-cylinder.price than the information contained in vehicle.boat.price. Furthermore, if information were also contained in vehicle regarding price, then the system would infer that such information would be more relevant to vehicle.car.6-cylinder.price than information contained in vehicle.boat.price, not merely because the logical proximity is less, but also because vehicle is an ancestor of vehicle.car.6-cylinder.price, thus implying that information located there is likely to be of relevance to any of its descendants.

3 Criteria preference revisited

Once it has been decided to relax a particular term, the notion of criteria preference [2] is used to determine which relaxations are acceptable, and which amongst the acceptable relaxations are preferable. The result is a frequency distribution of relaxations that the user has indicated are acceptable to him for a given attribute. Whilst this method does provide a workable solution to determining criteria preference, it does blur some of the information which could be used to make such a determination more precise. As such a new method of generating criteria preference profiles (CPPs), which is tailored for the specific question for the specific user is presented here. There are two methods of directly inferring acceptable CPP of the user. Both processes are described in detail in [2]. A brief outline is included below.

The first involves obtaining information from observing which selections the user found acceptable. When a user asks a query, the initial criteria are recorded. When the user selects an answer as being acceptable, the system compares the original criteria in the query with the attribute values which yielded the acceptable choice. The deviation between the original and final criteria is recorded as having been an acceptable relaxation on one occurrence.

For example, consider the instance where a user enters the query

```
select vehicle
where type='boat' and colour='red'
    and price < 50,000
```

the system records the original criteria as being type='boat', colour='red' and price < 50,000. Suppose that after being presented with a set of relaxed answers, the user selects one for which the associated values are type='boat', colour='red' and price < 60,000. The system assumes that selection is a likely indicator of
acceptance, whilst being aware of the fact that people may select an option for reasons other than acceptability. The system compares these two sets of criteria, and records the fact that in this instance, a relaxation of vehicle.car.6-cylinder.price of +10,000 was acceptable. A frequency histogram is maintained for both categorical and numerical relaxation data.

The second method of inferring involves observing relaxations that the user themselves may make over the course of a series of queries. In a given search series, all criteria in queries after the first are compared with criteria in the initial query. If the range of the criteria is found to be larger than the range of the initial criteria for a particular term, then the user is said to have autorelaxed that term. The system takes note of these autorelaxations, as the user is explicitly indicating to the system a pattern of relaxation that they consider acceptable.

Consider the following queries.

Query 1: select vehicle
        where type='boat' and colour='red'
        and price < 50,000

Query 2: select vehicle
        where type='boat' and colour='red'
        or colour='white' and price < 45,000

Initially, the system observes that at t=0, the original criteria for the query were type='boat', colour='red' and price < 50,000. The system observes that at t=1, the user appears to be searching for the same thing (i.e. that the subsequent query appears to be a part of the same search). It further observes that the set of values which will satisfy the compound criteria for colour (i.e. colour='red' or colour='white') has increased, and thus the system infers that the criteria for colour has been relaxed by the user themselves. The system records that the user has indicated that a relaxation of vehicle.boat.colour from 'red' to 'white' is acceptable as they themselves have made this relaxation. The system also observes that the set of values which will satisfy the criteria for price has decreased (i.e. price < 45,000) and so this value has not been relaxed. Any subsequent relaxations are calculated as a deviation from the original value (i.e. price < 50,000) rather than any intermediate results.

The information obtained from both methods is combined into a single CPP. As a necessity of practical implementation, given storage limitations, information about numerical data is stored as ranges, rather than as precise values. The range chosen by the system is decided by the administrator of the system and the level of precision they require for their analysis.

![Price Deviation (thousands of dollars) vs Acceptable deviation](image)

Figure 2: Sample CPP

Figure 2 shows a sample CPP for a particular user. It indicates, for example, that 36 times the user has accepted a price deviation between $0-15,000 above the criteria that they specified. As such, it is used to guide which relaxations are acceptable and which are preferable to others.

This method will provide a good guide to relaxation when the system has already been able to observe the users behaviour for terms specifically appearing in the query. For example, if the user has made a number of queries dealing with "boat" and "price", and asks another question regarding "boat" and "price", then the above methods will help yield good results. However, the above methods are deficient in situations in which the user asks a query dealing with a new set of terms, even though they may have asked many other questions of the system previously. This is the situation where there is a considerable amount of information about the user within the profile, but none dealing with the specific query they are asking (if the system does not have a sufficient amount of knowledge about a specific user, then the use of stereotypes is applied). In this instance, the system attempts to see general patterns of behaviour of the user, and to map this to the current question being asked.

### 3.1 Analysing behaviour

To deal with this deficiency, when a user asks a query which leads to a relaxation of a term that has no (or little) history contained in the profile, the system attempts to determine whether the user has asked any other questions which may be "similar" enough to be of use for this particular query. The node in the LAH representing this term is called the target node.
For example, suppose a user asks a query involving "motorcycle" and "price". Suppose also that the system currently holds no information regarding the users criteria preference for "motorcycle" and "price" (represented on the LAH by the node motorcycle.price), but does hold information about the users criteria preference with regards to "boat" and "price". As "motorcycle" is close to "boat" in the LAH, the system determines that the profile for boat.price may be relevant to motorcycle.price, and so will map the profile to motorcycle.price in this instance. The system also looks for general trends to a particular attribute (for example, does the user have a general profile for many instances of price) and the proximity on the graph of instances and the target node. The detailed processes by which this process takes place are further examined below.

3.2 Compare and cascade

The basic method for generating the profiles to be examined for similarity is one of "cascade and compare". The system locates the term to be relaxed within the LAH. It then searches for all other occurrences of the entity name within the LAH. So, considering our example above, when the system comes to relax "price", and there is no applicable profile for motorcycle.price, it will search the LAH and identify all other occurrences of all attributes named "price". Starting at each entity, the system generates a CPP for the entity in it's given context (the precision of which will depend on the degree of detail included in the LAH). For each CPP, care is taken only to use the instances where the entity in question has been relaxed in the context of the node in the LAH in which it appears. Each CPP has a degree of certainty (DC), which is a function of the number of relevant instances that went into generating the CPP. The more instances that went into generating the CPP, the higher the degree of certainty that this CPP accurately reflects a general pattern for the user. Two CPPs will only be compared if \( \text{MIN}(\text{DC1}, \text{DC2}) > T \), where T is the threshold population required to enable a meaningful comparison of CPPs.

Having generated all CPPs, the system will then compare them, and attempt to "cascade" them upwards if appropriate.

3.3 Comparison

In order to compare two CPPs, the system first normalises the appropriate CPPs, vectorises them and then compares these vectors. The number of dimensions requiring normalisation depends on the type of information that the attribute holds. For categorical data, only the frequency dimension needs to be normalised. For numerical data, both the frequency and numerical dimension need to be normalised.

Normalising the frequency dimension involves changing this dimension (the y-axis in Figure 2) from an absolute measure to a relative one, so that the frequency dimension becomes a measure of the percentage that a particular value occurs for the given set of data.

Normalising the numerical dimension involves more work. The first and easiest thing is that the values of the various ranges are discarded and replaced by simple counter functions. Thus, in the normalisation of the CPP represented in Figure 2, the value of +(0-15) would be replaced by +1. We are interested here in a general pattern of responding to an attribute, rather than the specific values involved in such a response. If the CPPs to be compared contain the same number of divisions, then the system can proceed to vectorisation. Otherwise, it must normalise the number of divisions between the two graphs. the CPP with the highest number of divisions is given the same number of divisions as the one it is to be compared to. The new values for the frequency are calculated according to the following equation:

\[
f_i = \alpha_1 + \alpha_2 + \alpha_3
\]

\[
\alpha_1 = \begin{cases} 
\frac{(n - i - 1)}{m} f_p, & \frac{p - 1}{n} \neq \frac{i - 1}{m} \\
\frac{p}{n} & \frac{p - 1}{n} = \frac{i - 1}{m}
\end{cases}
\]

\[
\alpha_2 = \begin{cases} 
\frac{\frac{1}{m} - \frac{q}{n}}{f_q}, & \frac{q}{n} \neq \frac{1}{m} \\
\frac{q}{n} & \frac{q}{n} = \frac{1}{m}
\end{cases}
\]

\[
\alpha_3 = \begin{cases} 
\sum_{i=p+1}^{n-1} f_j, & q > p + 1 \\
0 & q \leq p + 1
\end{cases}
\]

where \( n, m \) are the number of divisions contained in a histogram, \( n > m, p \in \mathbb{N} \) such that \( p/n \geq (i - 1)/m \) and \( \forall y [y \in \mathbb{N} | y/n < (i - 1)/m \text{ or } y \geq p], q \in \mathbb{N} \) such that \( q/n \geq i/m \) and \( \forall y [y \in \mathbb{N} | y/n < i/m \text{ or } y \geq q] \). \( f_i \) denotes the frequency recorded at division \( i \) at the histogram.

Figure 3 shows the results of a normalised CPP. The resulting normalised CPP is then vectorised by recording the resultant relative frequencies in adjacent cells in the vector. In the above example, the resulting vector for the histogram would be \( V_1 = [5, 13, 24, 31, 11, 16] \).

After both CPPs are vectorised, the Difference Vector (DV) is then calculated as \( V_1 - V_2 \). In the above example, consider that \( V_2 = [3, 15, 26, 29, 10, 17] \). Thus,
Figure 3: A Normalized CPP

$DV = [2, -2, -2, 2, 1, -1]$. The DV is then compared with the Similarity Tolerance Vector (STM). The absolute value of each term in the DV is compared with the corresponding term in the STM in order to establish whether these two CPPs are similar. Thus, if the STM = [2, 2, 2, 2, 2, 2], then the two CPPs described above would be considered as similar.

When two or more sibling CPPs are similar, the system generates a composite CPP of all similar grandchild CPPs, and stores this at the grandparent node along with the grandchildren that comprised the composite CPP. The grandparent node may contain more than one group of similar CPPs, along with a path of applicability, which indicates which grandchildren comprised the composite CPP. The choice to include multiple composite CPPs ensures that nodes with few descendants will not have dominate from nodes with more descendants, where the probability of finding a single CPP to fit all profiles diminishes as the number of descendant nodes increases. The composite CPP so formed is an indication that the CPPs provide a general pattern for acceptable relaxations for the grandchildren of that node. Such a CPP is said to have a wider applicability than the CPPs of the grandchildren.

### 3.4 Choosing the appropriate CPP

The system next attempts to determine the appropriate CPP to map to the target node. At this stage there are 2 major considerations for the system. CPPs near the target node are likely to be more relevant to the target node. However, CPPs with a wider applicability that cover the grandparent of the target node are also likely to be more relevant, as the wider applicability suggests that this is a more general pattern which the user widely holds.

In order to calculate which CPP should be mapped to the target node, the system calculates a relevance factor (RF) for each CPP. The CPP with the highest RF is then assigned to the target node for the purposes of this calculation.

$$RF = (10 + w1 + w2)LD/CN$$

where LD is the logical distance from the CPP to the target node, CN is the total number of nodes on the path of applicability, $w1$ is the weighting due to the logical proximity of the CPP to the target node, and $w2$ is the weighting due to the applicability of the CPP. The CPP with the highest RF is assigned to the target node.

Initially, $w1=w2=0$. $w1$ and $w2$ are used as indicators so that the users preference for either logical proximity or applicability can be taken into account. When a user actually makes a selection, the relevant term value is compared to the CPP actually assigned to the target node. The system calculates the probability of the term value that the user has chosen appearing for the CPP assigned to the target node, and the CPP of the node with the widest applicability that is relevant to the grandparent of the target node, and the CPPs of the grandsiblings of the target node. If the highest probability was for one of the nodes with the smallest logical proximity and the assigned CPP was close to the target node, this is taken to indicate that the system is behaving according to user preferences and no action is taken. Similarly, if the node with the highest probability was that with the widest applicability and the assigned CPP was close to the target node, this is taken to indicate that the system is behaving according to user preferences and no action is taken. If, however, the node with the highest probability was that with the widest applicability and the assigned CPP was close to the target node, the system infers that the users preferences are not being met in this case, and so $w1$ is increased by 1. A similar mechanism exists for increasing $w2$.

Continuing our example, let us consider that after the compare and cascade algorithm has finished, the system decides to assign the composite CPP from "entity" to vehicle.motorcycle.price. The user is then presented with a list of possible answers and selects one which has a price of $10,000, whereas his original criteria was price < 7,000. The system calculates that the probability of accepting a price deviation of +3,000 (converted to a normalised value) is highest for the CPP of vehicle.boat.price. Thus the CPP actually assigned to the node is not the one that best suggests the value that was actually found acceptable to the user. In this case, the system would increase $w1$ by 1, indi-
cating that in this instance the CPP with a smaller logical proximity was more relevant to the user than that with wider applicability. Thus the system can learn and adapt so that inferences it makes regarding the user will be more finely tuned to the requirements of that specific individual.

4 Importance revisited

The notion of importance as proposed in [2] maintained that the level of importance of a term stood in an inverse relationship to the order of relaxation. The determination of importance was based on psychological findings that suggested that, on the average, terms placed earlier in a criteria statement were more important than those appearing later in the statement. Whilst this solution is good for general cases, it does not take account of individual differences within the population, nor does it cater for people whose representation of importance does not conform to the norm. It also fails to take into account differing patterns of language use that may arise from differing cultural or linguistic backgrounds (such as where English is not the native language, or where the query is being entered in a language other than English). The authors suggest a method for re-evaluating the method by which importance should be determined for each individual, to take account of this problem.

It is proposed that initially the system still use the original method to determine the importance of terms to a user. As the system learns more about a particular user, the system can the re-evaluate it’s assumptions (whether this be the default assumption or later derived ones) as to the method by which importance should be determined for this particular user.

The notion of importance is used to determine the order of term relaxation. The least important term is relaxed first. The job of the system is therefore to determine the ordinality of importance of terms within a given query.

One excellent source for such information is by looking at autorelaxations of the user. By examining which terms the user themselves relaxes first, the system can attempt to determine the method by which this specific user signifies importance. The system can search through the autorelaxations of the user, and attempt to determine any patterns which emerge which indicate importance for the user. When examining a particular autorelaxation series, the system defines the least important term as the one which is relaxed first, the next least important term as the one that is relaxed second, and so on. The system can look for a number of indicators in an attempt to derive a pattern which may indicate how the users importance preference is expressed in their query.

The easiest indicator to look for is the serial position within a query. For example, is the fist attribute always the most important, or the last? If such is found to be the case, then the system can easily modify the way it determines importance for that user. However, if such is not the case, there are other methods of investigation the system can employ.

The system can attempt to look for context dependent patterns of importance. If the system determines that a method for determining importance, for example serial positioning of attributes, is not valid for a particular user, then it can attempt to determine if there are any other patterns which suggest importance. Is a particular term always more important than others? Or are attributes which deal with a particular subject always more important than others (this would be reflected by the fact that the most important attributes were always children of some ancestor node in the LAH ... for example, one might find that anything pertaining to "vehicles" was always more important to a user than anything else). Are there certain combinations of attributes which suggest importance? Or is there one specific value for an attribute that always signifies importance?

The system compares and decides amongst competing strategies by building an Importance Decision tree. The Decision tree is a copy of the LAH, with each node containing an Importance Decision Strategy (IDS) instead of a CPP. The IDS is a frequency histogram of the importance of the relevant term, represented as an ordinal index (1 meaning most important, 2 meaning next important etc). IDSs are cascaded upwards using the same compare and cascade algorithm as applied for determining CPPs. Just as with a node containing a CPP, each IDS contains a degree of certainty, a path of applicability and a level of applicability.

After applying the compare and cascade algorithm, the system can infer the best IDS for a target node in a fashion similar to choosing the best CPP for criteria preference.

As such, the system will learn over time the best method of determining importance for each specific user. Rather than constraining a user to the predefined statistical norms which fail to take cultural and linguistic differences into account, the system can employ a strategy for determining importance which is optimal for each individual user.
5 Summary

The idea has been proposed to guide query relaxation in ways relevant to each specific user by modelling their behaviour and using this to try and ensure that the relaxation matches the users requirements. The proposed method had a number of deficiencies, mainly in its inability to deal with situations with which it had no prior experience, as well as a rigid rule for determining importance which did not take into account individual, cultural or linguistic differences. This paper proposes some solutions to these problems. Specifically, it is proposed that stereotypes be used initially for new users, so that the system has some information upon which to base its relaxations, and that these stereotypes become less important the more the system learns about the individual. A method is also proposed for inferring an appropriate relaxation strategy where there is no recorded history for the exact query being asked, but where the system contains information of previous queries which were similar to the current one in some fashion. Finally, a method is proposed whereby the system observes and re-evaluates the strategy it uses to determine importance, rather than adhering to only one, rigid rule, so that the determination of strategy will be appropriate to the specific individual using the system. It is hoped that with these improvements, the original system will be more able to fulfil its aim of providing each user with a set of answers which is optimal to each individual.

References


