Abstract

The project addresses the problem of "unskilled" human request expression and assimilation. The same phenomenon, the same object may be described in different ways by different people. Further, different recipients can interpret the same sentence differently. Orthographic or grammatical errors make the problem even worse. This is the main barrier for development of any system for recognizing the meaning of human speech, text, or information search request in particular. As the result, the most of current data base interfaces leave the step of the request definition to the users. This drastically limits the data mining effectiveness for them.

The originality of the proposed approach is based on designing a request specification interface with direct assistance functionality. It helps client to describe his request absolutely correctly being sure the system "understands" him completely. As the systems reaction the client may expect fruitful data mining results every time.

1 Introduction

In this paper we would like to focus our study on the problem of "unskilled" user access to the data base systems.

The user interface is a tool for interpretation of the user information or data search request into the data mining query. The "unskilled" user is a person having neither information search experience, nor special knowledge on the data he looks for.

Needless to say, the user interface is a key part of any data base system, especially in the cases assuming the general public access. The entire system quality, as well as its usefulness, is evaluated by the quality of the interface.
There are many criteria of the interface evaluation. From the point of view of the "unskilled" user, the interface must be simple and easy to use, and it must provide productive result of the data/information search. Definitely, the search result strongly depends on the request specification quality; it is difficult to expect productive result in case of the incorrect request. It means the interface should exclude anyhow or reduce to the minimum the possibility of specifying incorrect requests.

The final goal of the present paper is to show a newly developed approach for the interface design and implementation, which can guarantee correct search request specification regardless of the user experience and skill.

2 Problem definition

2.1 Simple and easy request input

Certainly the main criterion for the user interface is the simplicity and easiness of the request specification. Let's consider some approaches to the user interface design dominated nowadays from the point of view of "unskilled" user.

The most popular method of the request specification is so called Free-Text approach. In this method user specifies his request in an arbitrary form like it is shown in Fig. 1. The user interface analyzes the text structure, text meaning, and interprets it into the search query. Certainly, this method is very simple and easy. However, this is well known that even the same phenomenon, the same object may be described in different ways by different people. Further, different recipients can interpret the same sentence differently. Orthographic or grammatical errors make the problem even worse. Thus, the interpretation of the request text by the user interface may be ambiguous and it may result in wrong search result, or even absence of any result.

The next method, called Push-Button approach, is one of the model-based methods. The user interface consists of the set of key words (Fig. 2), which fully covers the variety of the feature attributes existing in the database. In order to
specify the request the user needs to select a combination of the appropriate buttons. In this approach the interface is designed to exclude opportunities for having grammatical and/or orthographical mistakes in the request, and to increase the level of the user-system mutual understanding. However, it does not exclude the opportunity of making incorrect request completely. Even having a set, where each of the attributes individually exists in the database, an arbitrary combination of the attributes may be nonexistent. It means that some of the user search requests specified with the help of the Push-Button interface may produce no result.

2.2 Correct request input

It is very difficult sometimes even for an experienced user to find mistake in his request and to decide the further steps, when the search system reacts unexpectedly listing wrong data or showing no data. The request may be even correct, but just the particular database may not have the requested data or information. This is why it is so important to have request input methods preventing unexpected search results.

There is a group of methods, where, in order to avoid the attribute logic inconsistency, the features are divided into hierarchical sets (see Fig. 3). The feature hierarchy represents the data logic; the next group of available features is set and shown to the user depending on the previous selections. As the result, the selected combination of features is always correct, and the search is always productive. The sequence of the feature groups is preset by the interface; this is the main disadvantage of the method. Sometimes the use wants to specify the features in different order that is impossible in this method.

Figure 3: Hierarchical request input.
2.3 Summary

From analysis of the methods described above it was concluded that in order to develop an interface useful for the "unskilled" user a new method for the request specification is required. The method must combine the simplicity, easiness, and request input freedom of the Push-Button approach and ability of the incorrect requests excluding of the hierarchical methods.

3 Method description

3.1 General approach

It has no any technical difficulty to implement the Push-Button user interface outlook (see Fig. 2). The problem is how to prevent the input of inconsistent features combination with such interface. A reasonable answer is the interface system must analyze every selection of the user and exclude from the input list the features, the further selection of which may cause conflicts or bring no search result, like it is shown in Fig. 4.

Analysis of the feature selection can be implemented on the server side analyzing the database structure and possible inconsistency after every click of the user. Needless to say, this is the time consuming way because the interface reaction in this case depends on network capacity and speed between user and server sides.

Alternative way is to send user the data logic information together with the interface page, so the input analysis may be done on the user side without connecting the server after every act of the request specification. This approach looks more practical and was chosen for implementation.

Figure 4: Example of the interface reaction after attribute selection.
### 3.2 Data restructuring

The proposed method of request specification and processing is based on the idea of database structure transformation. The new data structure becomes more compact and "transportable", and it is more suited to the data analysis by the features or attributes.

As the rule, data in databases are structured in table form. Usually, a set of items is presented in a table, and the item name or number play the role of the key parameter of the set (Fig. 5). This data structure came from traditional archives and quite convenient for the database filling and maintenance. However, analysis or search of the data in such table by specific attributes and features is quite difficult.

Assuming that the "unskilled" user usually searches data in a database by features and attributes, it is more preferable to present the data set via the attributes, like it is shown in Fig. 6. In the new structure the attributes become the key parameter, and existence of any attribute in an item specification is described in bit-map relationship form, where one means the item has the appropriate attribute, and zero - the item does not include the attribute. Data in this structure is quite easy to analyze by the features. If any feature is selected, the appropriate row in the table has complete information on what items have the specified feature.

Bit-map presentation of the data structure becomes more compact than the traditional one. The data structure in this form can be easily and quickly loaded on the user computer together with the interface page for data search.

![Figure 5: Typical presentation of data in database.](image1)

![Figure 6: Bit-map relationship.](image2)
3.3 Request specification and processing

The request specification procedure consists of a sequence of attributes selecting and/or deselecting acts. Each individual act of selecting/deselecting is processed by the interface system and attributes inconsistent to the selections are defined.

Generally speaking, the inconsistency of any attribute may be defined by comparing the row (bit-vector, hereafter) of the bit-map corresponding to the attribute with the vectors of the previously selected attributes. It has been mentioned the vectors include information on what items in the database have the appropriate attribute. If any attribute does not exist in the set of items defined by previously selected attributes, it means this attribute is inconsistent to the selection and must be excluded from the list of attributes available for the further selections.

The inconsistency definition process can be described in terms of the Boolean algebra.

Let define a vector of the bit-map as \( V_i \), where \( i \) is the vector or the corresponding attribute number in the bit-map. Each element of the vector is a bit corresponding to the appropriate item; size of the vector is the number of items in the database. True value of the vector element means the attribute is included in the item specification; False - the attribute is not included. In addition, let introduce two more bit vectors; attribute selection status vector \( S \) and attribute availability vector \( A \). Size of the vectors is the number of attributes. True value of an element of \( S \) means the attribute corresponding to the element is selected. True value of an element of \( A \) means the attribute corresponding to the element is available for the further selection.

Having this definition in mind the inconsistency (or availability of \( i \) attribute) definition may be presented as:

\[
A(i) = \{ V_i \ AND \ A(S(j) \ AND \ V_j) \}
\]

Here AND is the logic AND; logic operation of a bit scalar with a bit vector means the operation of the bit with each element of the vector, the result is vector; logic operation of two bit vectors means the operation of each element of one vector with the same element of another vector, the result is vector; \( A(\text{vector}) \) is taken to be the logic AND of all the vectors in the set; \( \{\text{vector}\} \) represent the logic AND of all elements of the vector.

If the result is False the attribute must be excluded from the set available for further selection. It prevents specification of the incorrect request. If user finds that a desired attribute has become unavailable, he can make steps back in his selections and specify the attributes the most wanted first.

Thus, the method and user interfaces developed on the basis assists the user on the way of the request specification showing him possible inconsistency between the attributes.
4 Conclusion

New method for the data search request specification is designed and developed. The method assists user during the request specification process showing all possible inconsistencies between available features or attributes. It does not allow user to specify any incorrect request, which may produce wrong search result, or no result.

Several interfaces has been successfully implemented based on the proposed method for different application areas, namely for Metal Cutting Tools database, Real Estate database, Automotive database. Results of testing of the interfaces demonstrated the easiness, simplicity, and convenience of the new request specification approach.