Multimodal Database Query

Nicholas J. Haddock
Hewlett-Packard Laboratories
Filton Road, Stoke Gifford,
Bristol BS12 6QZ, U.K.
njh@hpl.hp.co.uk

Abstract

The paper proposes a multimodal interface for a real sales database application. We show how natural language processing may be integrated with a visual, direct manipulation method of database query, to produce a user interface which supports a flexible form of query specification, provides implicit guidance about the coverage of the linguistic component, and allows more focused discourse reference.

Introduction

Recently there has been a burgeoning of interest in the combination of natural language processing with visual and gestural forms of communication. The range of research includes the interpretation of combined linguistic and diagrammatic input (e.g. Klein and Pineda, 1990), the generation of multimedia explanations (e.g. Wahlster et al., 1991), the integration of NLP with hypertext (Stock, 1991), and the combination of natural language input with pointing (e.g. Kobsa et al., 1986), menus (Tennant et al., 1983), and forms (Cohen et al., 1989). The rationale for most of this work is that since the different modes are best suited to expressing different kinds of information, the expressiveness of the communication can be increased by employing a combination of modes rather than one in isolation. At one end of the spectrum, this rationale has led to applications in which there is a clear dividing line between the function of the different modes; for example, in GRAFLOG (Klein and Pineda, 1990), the function of linguistic utterances like *This line is a wall* is to provide real-world interpretations for the parts of a line drawing. Our own work lies at the other end of the spectrum. We are interested in exploring the power of combining natural language and direct manipulation when the function and expressive power of the two modes are similar.

The present paper describes this dual approach in the context of a specific, real database query application. We have integrated NLP with a visual, direct manipulation method of database query, in such a way that both query modes can express approximately the same range of queries. Our objective in this work is to explore the ways in which the copresence of a direct manipulation interface improves the usability and flexibility of the natural language interface. The concern of this paper is in the nature of our computational proposals, rather than empirical evidence about their utility. So far we have performed a user trial on the direct manipulation interface alone (Frohlich, 1991), and we intend to perform similar experiments with the combined interface and a stand-alone version of the NL interface.

We start with an overview of our application area, and then summarise the pertinent features of our direct manipulation interface. The body of the paper illustrates how we have integrated NLP with this interface, with respect to the available vocabularly, the portrayal of user queries, and reference to past queries.

Application Domain

The target of our application is a relational database used by a large UK company to summarise the value of their product sales. The main users of the data are sales professionals and their secretarial assistants. To date, these users have had two routes of access to the data: they can retrieve fixed-format financial tables using a menu-based query system, or they can use a rather brittle natural language interface (supplied by a third-party vendor). The design of our system stems partly from users' experience of these existing interfaces. Our user interface is geared to supporting the following user requirements: it should be simple to use; it should support flexible-format views or "reports" on the data; and it should allow new queries to be composed with reference to past queries.

Although our user interface is a prototype, and is not yet in use, we have done nothing to change the structure of the underlying relational database. The database summarises sales along three dimensions, or parameters: the product sold, the purchaser (in this case, retail outlets), and the time of sale. In conceptual terms, each dimension forms a hierarchy. For example, the leaves of the purchaser or "customer" hierarchy identify trading points (numeric identifiers corresponding to distinct physical retail stores); at the next level, these are grouped into

trading concerns (corresponding to the familiar high-street names of retail chains); and at the top level the trading concerns are grouped into corporate concerns (corresponding to the public limited company which owns the chain). Similarly, the time hierarchy represents financial time periods from leaf "bookweeks" up to the financial year, and the product hierarchy represents specific product box sizes up to a general classification of market areas.

Direct Manipulation Interface

The direct manipulation interface represents each sales dimension by a visual *domain model*. For instance, the temporal domain model (see Figure 1) is depicted as a scrollable timeline, demarcated into hierarchical time periods. The model presents both the *concepts* (e.g. "year", "quarter") and *values* (e.g. "1990", "Q2-90") which are distinguished in the temporal dimension. The product and customer domain models are similarly displayed as hierarchical "pick-lists".

Figure 1: Temporal domain model

Users pose queries by constructing the format, or appearance, of the report they want to see, using a technique we have dubbed "Query by Format". The present system supports three types of query in this fashion, providing: numeric summaries of sales with respect to specific parameters (e.g. the value of sales of Krunchy in June), lists of values related to the sales parameters (e.g. all trading points which bought Krunchy), and details on specific values (e.g. the telephone number of trading point 647). To simplify the exposition, the remainder of this paper will discuss only the first kind of query, requesting summaries of sales.

Figure 2 shows a sales summary table which the user has created and then evaluated as a query against the database. It represents the sales of Krunchy for Oct-90 and for Nov-90, for the stores Andrews and Walkers. This table is created by first selecting "Create a sales table" from a menu, which produces a skeletal table structure without row or column headings. The user then specifies the headings by gesturally selecting value elements (at any level) from the domain models and dropping them into appropriate positions on the table. Once the user is satisfied with the format of the table, it is submitted to the system where it is interpreted as a query against the database, and the derived results filled in. The

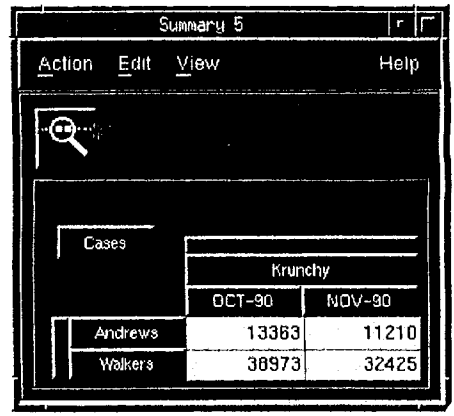

Figure 2: Tabular sales query

table therefore has a standard "intersective" semantics, in which each cell represents the summed total of sales with respect to its corresponding row and column constraints—for example, the top left-hand cell in Figure 2 represents the total value of sales of Krunchy in Oct-90 to Andrews.

Each such report created by the user forms a distinct window on the screen, and is therefore subject to standard window management functions such as iconisation. Importantly, the user can return to an existing report at any stage and refine it to create a new view—by adding, expanding, deleting, or replacing report headings.

Integrated Natural Language Processing

Natural language processing is based on the Core Language Engine (CLE; Alshawi et al. 1989), which performs application-independent processing from string segmentation and morphological analysis through to quantifier scoping and discourse reference, and produces semantic logical forms as output. The CLE contains "hooks" for application-specific modules, and we have used these to augment the CLE with an application-specific lexicon, a set of rules for reference resolution in our domain, and a module for evaluating the logical forms against the relational database. The present coverage of the NL system is similar to the gestural interface, in terms of the subject of the query and the conceptual parameters of the query. So we can ask questions about the value of sales with respect to any of the three sales parameters (e.g. *Show the sales of Krunchy 250g for Q3-90*) and questions about the sales parameters themselves (e.g. *What Andrews trading points are there?*). It goes beyond the graphics in supporting certain forms

of request which yield a simple textual response (such as yes/no questions). At present, natural language queries and direct manipulation queries have separate routes of access to the database, and we achieve the integration described below by translating between the two worlds at certain points in processing.

The following sections present three ways in which we have explored the integration of natural language with the direct manipulation interface. In each case we argue that the integrated interface is superior to stand-alone, teletype-style natural language interaction.

Vocabulary

A well-known problem with natural language interfaces to databases is that the user may be uncertain about the conceptual scope of the database and the supported linguistic coverage (Hendrix, 1982). The graphical environment of our NL interface offers a partial solution to this problem, since the displayed domain models remind the user of the linguistically available parameters of a sale. In particular, each domain model communicates the range of sales-parameter concepts and values that can be referred to, and in doing so shows one way of expressing the related content nominals in linguistic input. For example, the temporal domain model indicates that the underlined forms in *Get Krunchy sales for bweek 18-19 May-90* are available as lexical expressions. (We also allow for synonyms in the NL, given the tendency to refer to "bweek", say, as "week".)

Note that the domain models are an appropriate site for lexical reference, since they abstract away from the internal structure and content of the database in order to provide a user-oriented "view" of the data. For example, the temporal model depicts information from relational tables as a single hierarchy, and combines distinct database fields to form single values within this hierarchy which are meaningful to the end-user.

Representation of Queries

The user's natural language dialogue with the system is displayed in a separate window in teletype form. Yes/no and "how many" questions elicit a simple textual response, whereas the response to other queries is a textual pointer (e.g. *See table 13*) to a report displayed elsewhere on the screen. The decision on presentation style is made by a set of rules indexed on the sentence form (e.g. WH-question, imperative) and the requested class of data.

Here we will consider those queries which request sales figures. Figure 3 shows a numeric summary table that has been produced in response to the input *Show all sales in Nov-90 to Andrews trading points*. This linguistically created report is an "live" graphical object which has exactly the same tabular semantics as if it had been constructed by direct manipulation. In fact, we can think of the table as a representation of the natural language query in the language of tabular queries. The table readily expresses the linguistic constraint *in Nov-90* with the tabular label "Nov-90". The intensional expression *Andrews trading points* cannot be represented directly, since our tabular restrictions must be extensional values from the domain models and not intensional concepts. However, we can represent this intensional expression indirectly in terms of its semantically equivalent extension—i.e. the set of all trading points related to Andrews.

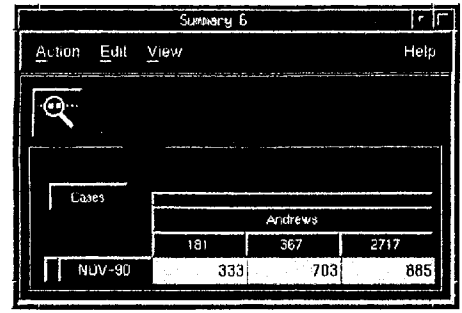

Figure 3: Tabular representation of *Show all sales in Nov-90 to Andrews trading points*

To display a natural language query and database response in this form, we must therefore not only retrieve the correct values from the database, but also generate row and/or column labels which correctly define the values in the table.[1] In general, this is a non-trivial task which ultimately requires sensitivity to the given/new information structure of the input; our approach uses information derived only from the logical form of the query, and corresponding values from the database. In the current version of our system, table labels are generated by the module which evaluates CLE logical forms against the database, as it searches for sales values in accordance with the constraints of the query. For example, to find *all sales in Nov-90 to Andrews trading points*, it searches for all sales values such that the following constraints hold:[2]

$$bmonth = \mathsf{nov} - 90$$
$$trading\_concern = \mathsf{andrews}$$
$$trading\_point = t$$

Here t is a free variable that will match any trading point value. Each time we find a matching sale value, we record, with this value, the corresponding values of the attributes *bmonth*, *trading_concern*, and *trading_point*. This results in a set of tuples of the form

[1] An alternative approach is simply to generate the tabular constraints, and rely on tabular query processing to produce the sales totals. We have not explored this approach.

[2] Note that the notation here has been simplified for the purposes of exposition.

<*sale_value*, *bmonth*, *trading_concern*, *trading_point*>, such as:

< 333,nov-90,andrews, 181 >
< 703,nov-90,andrews, 367 >
< 885,nov-90,andrews, 2717 >

if 181, 367, and 2717 are the only Andrews trading points to have bought goods in November 1990. Such a set of tuples can then be transformed into a tree structure which removes the repetition of values apparent in the set of tuples. This tree structure corresponds to a correct labelling of the table, where each node represents a label.

This approach extends to the representation of a class of more complex expressions involving negation, coordination and quantification. For example, under the wide-scope reading of *sales to all trading concerns except Andrews and Walkers*, we find all trading concerns such that there is a sale with the following constraints:

trading_concern = t
$t \neq$ andrews
$t \neq$ walkers

Here we generate a set of tuples of the form <*sale_value*, *trading_concern*>, where *trading_concern* will vary over all concerns except the excluded stores.

As an example of a reading which we cannot represent in a tabular query, consider *Total the sales for Jan-90 and Feb-90*. Here we can express the reading in which two totals are required, mapping to a table with one cell for each month. But we cannot express the reading where the user would like to see a single cell table, corresponding to the summed sales of January and February, because such a query cannot be specified in the tabular language. Hence, this reading is blocked, because the rules which transform the set of tuples {<*sale_value*, *bmonth*>} into a tree structure do not allow distinct values (i.e. jan-90 and feb-90) to be represented by a single node in the label tree.

Hence it is not possible to represent all natural language queries in our simple intersective, tabular language, because the former can be much more expressive. However, our interface approach alleviates this problem to a good extent, since tabular sales summaries are just one of a variety of gestural query devices at our disposal (i.e. those mentioned earlier in the paper) to express the communicative content of a natural language query, and we can add others as the need arises. None of our gestural query methods have the expressive power of a relational query language such as, say, QBE (Zloof, 1975); rather, we have created a set of graphical access methods tailored to our target users' needs, which strike a balance between expressive power and ease of use. We decided on the range of our query devices by analysing the transcripts of the users' real sessions with the existing natural language front-end, in addition to other forms of analysis, such as interviews with target users.

Given that natural language and direct manipulation both yield the same tabular output, what is the advantage of supporting two modes of query, rather than one? First, the user can build-up a table using whichever mode or combination of modes is most productive. By combining the presentation of gestural and linguistic queries, a linguistically generated report can be refined and extended by the direct manipulation operations described earlier. In the following section we will see how natural language query can extend an existing table, completing the circle of mixed-mode dialogue. Many of the distinguishing and productive features of natural language documented by Walker (1989), such as coordination, negation and quantification, can be beneficially applied to the present task. So a user may start a query table with the request *Find sales for the first week in every month*, exploiting the rich quantificational structure of English to swiftly generate a set of labels that would take a good many point-and-click actions. The user may then wish to further parameterise this query with certain products which are best selected visually (perhaps because their spelling is difficult to remember).

Second, our user studies in this and other domains have shown that there are differences in the preferences of individual users. Some users simply prefer the feel of, say, natural language interaction, for reasons which are difficult to explicate with theoretical factors such as those above. Here the user may issue even gesturally simple commands like *Open* from the linguistic command line.

Reference to Past Queries

Given the exposition so far, we can construct a new table with either natural language or gestures, but can only modify an existing table using direct manipulation. To complete this picture of multimodal dialogue, we must also allow linguistic queries to refer to past tables and from them specify modified versions.

In our application domain, we expect users to have several reports open on their screen at any one time, and identifying an individual table solely in terms of the content of a referring expression can be arduous, if not impossible. One option is to name the report using its unique identifier. In common with many other systems (e.g. Kobsa et al., 1986), we also allow the user to refer to an object by pointing to it. Our present system only allows one such deictic reference per sentence, and behaves as follows. At any point, the user can click on a button on a table to make it the "context" for the next linguistic query. If this action occurs, an anaphoric referring expression, like *these Andrews sales* in *Show these Andrews sales as a bar chart*, is then taken to refer *directly to* the contextual

table, assuming that the content of the expression (in this case, *Andrews sales*) does not contradict the content of the table. However, if the context button is pressed and then followed by the use of a definite noun phrase, as in *Show the sales for Jan-90*, then the table is seen as providing the *universe of reference* (i.e. the set of sales specified by the table) for *the sales for Jan-90*, rather than the referent itself. In this case, then, the query will yield a new table which combines the constraints of the contextual table with the Jan-90 constraint provided by the definite NP.

For completeness, the primary objects arising in the discourse are tracked using the CLE's discourse model, which is based on a salience-ranking view of discourse reference. We track all tables that the user has created and evaluated, whether through direct manipulation or natural language, rather than just those arising from the linguistic dialogue. If a sales table is uniquely salient (because, say, it was the most recently created) in the discourse model, then an anaphoric expression such as *these sales* will be taken as referring to this table (without the need for pointing), and the query *Which of these sales is for Walkers?* will accordingly produce a new table extended with the label "Walkers".[3]

In the future we intend to experiment with schemes where more general gestural actions affect the salience of objects in the discourse model. For example, "lowering" or iconising table windows on the screen could reduce their salience rating, whereas "raising" them would increase it. Although this is an over-simplified view of how to track the user's focus of attention, such schemes would give the user the potential for more explicit control over the working set of objects available for reference.[4] This scheme, and our implemented treatment of multimodal reference, therefore support a flexible form of discourse reference which is more unnatural to attain in teletype-style linguistic dialogue.

Conclusion

The previous sections have explored the consequences of a thorough integration of a gestural method of query and natural language query in the context of a specific database application. We considered the case where the coverage of the two styles is similar, in terms of the range of expressible queries, and have demonstrated that several benefits acrue from this integration. First, by translating a user's NL query into a graphical query, we support a flexible approach to the specification of the query, in which the user can employ whichever combination of modes is best suited to the "micro" tasks involved in query specification. Second, the visual interface gives implicit guidance to the user as to the coverage of the natural language interface. Third, the use of direct manipulation to focus discourse reference offers a more flexible dialogue structure than found in a pure NL interface. In the future it would be profitable to empirically assess these present implemented proposals, and investigate further computational issues, such as the generation of linguistic descriptions of gestural actions.

[3] Currently, language can add but not alter table labels, as would be required for the elliptical reading in which Walkers replaces an existing store name.

[4] Cohen et al. (1989) advocate a similar technique in which the user can direct manipulate the visually displayed tree structure of the discourse. However, in our proposal the discourse structure is inherent in the visual layout of the screen, without the presentation of meta-level information about the dialogue.

Acknowledgements

Many of the ideas discussed in this paper were developed through conversations with Andrew Nelson, who also implemented the algorithm for mapping relations into minimal-node trees. The direct manipulation interface described here was implemented by Andrew Nelson and Steve Loughran. Thanks also to David Adger, Phil Stenton, David Frohlich and Lyn Walker.

References

Alshawi, H., Carter, D. M., van Eijck, J., Moore, R. C., Moran, D. B., Pereira, F. C. N., Pulman, S. G. and Smith, A. G. [1989] Research Programme in Natural Language Processing: Final Report. SRI Project No. 2989, SRI International Cambridge Computer Science Research Centre, Cambridge, U.K.

Cohen, P. R., Dalrymple, M., Moran, D. B., Pereira, F. C. N., Sullivan, J. W., Gargan, R. A., Schlossberg, J. L. and Tyler, S. W. [1989] Synergistic Use of Direct Manipulation and Natural Language. In *Procs. of CHI-89*, Austin, Texas, May 1989.

Frohlich, D. [1991] Evaluation of the SAMI System Prototype. Technical Report No. HPL-92-17, Hewlett-Packard Laboratories, Bristol, U.K.

Hendrix, G. [1982] Natural Language Interface. *Computational Linguistics*, Vol. 8, No. 2.

Klein, E. and Pineda, L. A. [1990] Semantics and Graphical Information. In *Procs. of INTERACT-90.*

Kobsa, A., Allgayer, J., Reddig, C., Reithinger, N., Schmauks, D., Harbusch, K., and Wahlster, W. [1986] Combining Deictic Gestures and Natural Language for Referent Identification. In *Procs. of COLING-86*, Bonn, Germany.

Stock, O. [1991] Natural Language and the Exploration of an Information Space: the ALFresco Interactive System. In *Procs. of IJCAI-91*, Sydney, Australia.

Tennant, H., Ross, K., Saenz, R., Thompson, C., and Miller, J. [1983] Menu-based Natural Language Understanding. In *Procs. of 21st ACL*, Cambridge, Mass., June 1983.

Wahlster, W., Andre, E., Graf, W., and Rist, T. [1991] Designing Illustrated Texts. In *Procs. of 5th EACL*, Berlin, Germany, April 1991.

Walker, L. [1989] Natural Language in a Desktop Environment. In *Procs. of HCI International '89*, Boston, Mass.

Zloof, M.. [1975] Query by Example. In *Procs. of AFIPS 44 NCC*, 1975.