Integrating an Intelligent Interface with a Relational Database for Two Way Man-Machine Communication

Michael Anderson
Dong-Guk Shin

Computer Science and Engineering Department
University of Connecticut

Abstract

This paper is concerned with the design of a system capable of handling ad hoc user responses to system initiated questions. To this end we have developed an approach called the expectation-driven response understanding paradigm. This paradigm assumes that each system generated question is accompanied by an expectation of what the user should respond. The system attempts to link this expectation with the actual user's response in order to quantify its appropriateness in relation to the system's original question.

A prototype system based on this paradigm has been implemented and integrated with the INGRES database system. In this prototype, the system issues a question to the user, accepts a response, and decides the appropriateness of this response using common sense knowledge stored in a knowledge base and data stored in INGRES. Results of testing substantiate the paradigm proposed.

1 Motivation and overview

There have been numerous attempts to integrate Natural Language Interfaces with conventional database systems. Unfortunately, most of these have failed to capture a significant share of the commercial market. One of the major reasons for this is that user's queries tend to be incomplete, ambiguous, vague, etc. and these interfaces are not capable of handling such imprecision. An obvious way to handle such problems is to design a system that can ask the user for clarification however allowing the system to engage in two-way communication requires a system with an ability to handle a wide variety of ad hoc responses from users.

This work is concerned with implementation of one of the most critical components needed to allow a system to engage in a two-way dialogue with a user. This component focuses on the ability to accept arbitrary responses from a user to system-initiated questions. Such questions could arise when the system needs clarification given incomplete, ambiguous, or imprecise input from the user [2][3][12] or simply as the result of the system seeking information from the user. Consider, for example, a system generated question such as Where did you do your thesis? The question might be answered in a number of ways:

At the University of Michigan.
At Ann Arbor.
At Boston.
At a state school.
My advisor was John Doe.
At McDonald's.

Ideally, the system should be able to handle all of the above responses and, further, it should be able to discern how completely its question has been answered. This flexibility seems to be achievable if the system is given knowledge about what it is expecting as an answer, common-sense knowledge of the world and related data, and an ability to make inferences based on this knowledge and data. A discussion for each of the above responses follows. It is assumed that the system is expecting the user to respond with a university name.

At the University of Michigan: The system should immediately understand that the query has been answered directly and fully with little or no inferencing.

At Ann Arbor: The system should determine that the expectation has not been met directly by the user's response and must search for a way to link this response with its expectation. The system should infer that the user is referring to the University of Michigan by using the knowledge that Ann Arbor is a city and universities are located in cities, and the database fact that the University of Michigan is the university located in Ann Arbor.

At Boston: The system should realize that it has received an ambiguous answer since more than one university exists in Boston. It is important to note that the system, even though given an answer that needs further clarification, still determines that the response provided is somehow related to the its question.

At a state school: Given a user response that maps into no appropriate answer but still bears some relation...
to the system's expectation, it would be desirable for the system to note the relationship and determine that, although the question has not been answered fully, some effort in that direction has been made. If no data is kept on the types of universities, the system might only be able to infer that the user meant *At a state university* and never come any closer to the expectation than this. Although the expectation is never fulfilled, the response does have something to do with it so the system deems that the question has been vaguely answered.

*My advisor was John Doe.* The system should be able to establish that John Doe is employed by a university and infer that the user is referring to that university. The inference chain might look something like,

- *My advisor was John Doe.*
- *My advisor was a professor named John Doe.*
- *My advisor was a teacher named John Doe.*
- *My advisor was a teacher named John Doe who was employed by a school.*
- *My advisor was a teacher named John Doe who was employed by a university.*
- *My advisor was a teacher named John Doe who was employed by the University of Michigan.*

At this point, the system should determine that the query has been answered, indirectly but fully. The knowledge and facts needed in this example include the relationships between advisor, professor, and teacher, the relationship between a school and a university, the common-sense knowledge that teachers are employees of schools, and the fact that John Doe is employed by the University of Michigan.

*At McDonald's:* The system must know when the response given to the query has no relationship to the expectation. All inferences that can be made from this response are sufficiently far from the given expectation that the system deems it as having nothing to do with the query.

The approach used to process this wide range of ad hoc responses is called the *expectation-driven response understanding paradigm* [10]. It assumes that every system-initiated question is accompanied by an expectation of what will it receive as an answer from the user and uses this expectation to measure the appropriateness of the user's actual response. This is accomplished by computing the *conceptual distance* between this expectation and response. This conceptual distance is computed using the knowledge base and relational database in an attempt to link the system's expectation with the user's response. The types of knowledge used and the number and nature of the inferences made all figure into this computation. The conceptual distance between the expectation and response can then be used to evaluate the appropriateness of the user's response.

## 2 $L_k$ knowledge representation

To alleviate the ambiguities of natural language and facilitate the symbolic manipulation of user responses, a more standardized internal representation of language is required by the system. To this end, $L_k$ [9] has been devised. $L_k$ is a knowledge representation formalism which attempts to combine the rigorousness of formal logic with the structural flexibility of frames [6]. The main structure of the language used in this proposal is the *concept term* or *c-term*. A c-term is a representation of some entity, event, or state. It has a *concept label* and can be followed by a *restriction*, which is itself a series of *concept-connector c-term* pairs (each pair called a *restrictor*). For example, the University of Michigan in the city of Ann Arbor could be represented as UNIVERSITY [location: CITY [val: ANN-ARBOR], val: UNIV-MICHIGAN]].

The process of inferencing is one of syntactically manipulating c-terms. Restrictors can be introduced or inherited and the c-terms they contain can themselves be manipulated, concept labels can be substituted for more specific or more general ones, c-terms can be imbedded inside other c-terms that have the appropriate constraints, and, furthermore, any c-term that is contained in the restriction of another c-term can be released to act as an autonomous c-term. These transformations are accomplished by the application of a number of inferencing rules, which are an integral component of $L_k$ itself.

- **Downward Label Substitution** allows a concept label of a c-term to be more specifically denoted.
- **Upward Label Substitution** allows a concept label of a c-term to be more generally denoted.
- **Membership Identification** allows a c-term to be identified as a member of a set.
- **Concept Connection Identification** allows a matching c-term to fill a restrictor of another c-term.
- **Restrictor Introduction** allows the free exchange of restrictors between c-terms with identical concept labels.
- **Restrictor Inheritance** allows a c-term to be additionally restricted by restrictors of related c-terms.
- **Restrictor Release** allows the promotion of c-term that is imbedded in the restriction of another c-term.

These seven transformation rules are augmented by **Database Consultation** which includes relevant facts from the database into the process by mapping concept labels of c-terms to data in the database.
3 Conceptual distance

We propose the notion of conceptual distance [10] as a means to quantify the sense of appropriateness a response has to a given question. The conceptual distance between two c-terms is defined as the path produced by the transformation processes of \( L_\beta \) that links one c-term to the other. It is a measure of how closely related these two c-terms are. We are most interested, in the case of expectation-driven response understanding, in the conceptual distance between the system's expectation and the user's response to that question. For example, the path between the system expectation UNIVERSITY[\text{val:*TARGET*}] (where *TARGET* is a variable denoting the information sought by the system) and the user response ANN-ARBOR might be

\[
\text{ANN-ARBOR, } \text{CITY[\text{val:ANN-ARBOR}], UNIVERSITY[location:CITY[\text{val:ANN-ARBOR}], val:UNIV-MICHIGAN].}
\]

Through various transformations and the use of both knowledge base information and database facts, the user's response has been successfully linked to the system's expectation. This linkage is deemed the conceptual distance between this expectation and response.

The process of computing the conceptual distance between an expectation and response requires a heuristically guided control structure to focus the attention of the system on the most relevant transformations. A heuristic ranking of a given transformation can be formed by combining a rating of the similarity of that transformation with respect to the system's expectation and the accumulated cost of deriving that transformation. This heuristic can then be used to drive a modified A* search algorithm [4] in its search for the best possible interpretation of the user's response in terms of the system's expectation.

When attempting to compare a user's response and it various transformations to the system's expectation, it is necessary to be able to categorize which parts of that response/transformation are actually similar to the expectation and how they are similar. To this end seven similarity metrics that describe possible relationships between c-terms are presented as follows, roughly in order from the most important similarity to the least.

Two c-terms are related via Head Concept Match if each has the same head concept. Two c-terms are related via Sub Concept Connection if the head concept of the first is a subset of the head concept of the second. Two c-terms are related via Super Concept Connection if the head concept of the first is a superset of the head concept of the second. If the first c-term matches the filler of restrictor in the restriction of the second c-term, or vice-versa, then these two c-terms are related via Concept Connector Connection. If a response or transformation has a restrictor with the same concept connector as the target restrictor in the expectation, it is related to the expectation via Target Restrictor Present. If the restrictions of two c-terms have no conflicting restrictors they are related via Restriction Match. If a response or transformation has a set of restrictors that match each of the non-target restrictors in the expectation, it is related to the expectation via All Non-target Restrictors Present.

Similarities pertain to portions of c-terms. When actually comparing two c-terms, we can expect that a number of these similarities may be present between the two. When dealing with the group of similarities between two c-terms we refer to this group as the similarity set of the two c-terms. The first four similarities appear to be mutually exclusive; that is, it doesn't seem likely that a pair of c-terms that is related in one of these four ways could also be related in any of the others, therefore, there are only forty similarity sets to consider. Given the rough ordering specified for the similarities and the relation of simple pattern matching [5] to similarity sets, a full ordering of the similarity sets seems possible [1]. This ordering, then, produces a ranking of the similarity of one c-term to another and can be used to guide the computation of conceptual distance.

The second requirement needed to compute the conceptual distance between two c-terms is a determination of the cost incurred in the transformation of one into the other. This cost is a measure of the plausibility of the inferences made during this transformation and is determined by the types and number of operations used to make it. Each operation is given a cost factor dependent upon the plausibility of the inference it makes, dividing operations into strongly plausible inferences and weakly plausible inferences. These costs are accumulated for the transformation in question and this sum is said to be the cost of this transformation.

4 Response evaluation categories

A user's response can be categorized in a number of ways. Five general classes seem to handle the bulk of possible responses.

A Direct Response is one in which the transformation process yields the fulfillment of the system's expectation within a system defined number of operations using
only "strong" operations.

An Indirect Response can then be defined as any response whose transformation process yields the fulfillment of the system's expectation but uses more than the number of operations allotted a Direct Response and/or uses one or more "weak" operations in this process.

An Ambiguous Response can be defined as a response that directly or indirectly fulfills the system's expectation with more than a single c-term.

A Vague Response is a response provided by a user to the system's query that, although does not fulfill the system's expectation directly or indirectly, none-the-less is deemed as having something to do with this expectation in a general sense via the ranking of its similarity set in relation to the expectation.

Finally, a Non Sequitur Response is a response whose transformation process never fulfills the system's expectation and whose best transformation's similarity set ranking in relation to the expectation is greater than the system defined cut-off point for a vague response.

5 System architecture

MIDMAN (Mixed-Initiative Dialogue MANager) is an ongoing, long-term project at the University of Connecticut that ultimately seeks to provide an intelligent user interface to a DBMS capable of engaging in dialogue with a user [1][7][8]. It is a software layer between a knowledge base and INGRES that attempts to fulfill this goal through the use of real-world meanings of the database facts stored in the knowledge base and the notion of conceptual distance. Such a system requires a collection of program modules designed to pose a query, generate an expectation, parse a response, compute the conceptual distance between this expectation and response, and evaluate the response to decide the next conversational move (Figure 1).

Ideally, the actual DBMS chosen, and its underlying model, should have no bearing on the interface proposed; MIDMAN is to be free of such constraints allowing for its use in many situations and domains. In fact, the portion of MIDMAN that this work deals with is now tied to the relational data model [11] and its
query language; in particular, it is serving as an interface to INGRES running on a Sun workstation.

This work introduces a simplistic knowledge base that includes only three types of knowledge: concept set membership, subset relationships between concepts, and concept constraints. These are sufficient for the testing of the current system but all the problems concerned with knowledge representation and knowledge base construction are pertinent here.

Two important modules have not been fully implemented in the current system. The Conversational Move Generator will be the main reasoning engine that will receive response evaluations from the evaluator and use this to decide what to say next in the dialogue. The Natural Language Interface will both parse user input into \( L_1 \) expressions and generate natural language from internal c-term representations of system queries.

The Conceptual Distance Calculator receives both the user's response and the system's expectation and attempts to connect them via knowledge in the KB, data in the DB, and the transformation rules. Its final output is data required by the Response Evaluator to make a judgement on the relationship of the user's response to the system's expectation. It's control structure is a modified \( A^* \) search algorithm that uses a heuristic based on the conceptual distance between any given transformation of the users response and the system's expectation to produce a path, if one exists, between this response and expectation. This path is optimal if there exists no other path between the user's response and the system's expectation that involves fewer transformations. The production of an optimal path using \( A^* \) searching requires that the heuristic ranking be the sum of 1) the cost of the transformation so far and 2) an underestimate of the cost of any transformation from this point. The first condition is met by the assignment of cost to each of the operations via a call to the Operation Cost Calculator. The second condition is not met as stated; this underestimate is simply not available. What is supplied is a ranking of the "closeness" of a transformation to the system's expectation via the Similarity Calculator, giving a goal-directed flavor to this algorithm. Although this does not strictly meet the second condition of the heuristic, the intent of this condition seems to be satisfied by this goal-directed quality. In reality, even though all paths produced during the testing of the current system were optimal, this modification may cause the algorithm to produce only near optimal paths when the system is using full-fledged data and knowledge bases. This loss of optimality does not seem to be a critical factor when the heuristic nature of such a system is considered. The system should be allowed some latitude in its understanding of a user's response just as human's are and even near optimal paths should most often prove to be acceptable.

\( A^* \) node expansion is provided by calls to the C-term Transformer which returns a set of transformations based on the current c-term. The search for an exact match with the expectation continues until 1) one is found with no ambiguity, 2) two are found indicating ambiguity, 3) the heuristic ranking of all unexplored transformations exceeds a given maximum, 4) the number of unexplored transformations exceeds a given maximum, or 5) there are no more unexplored transformations. At any such point, the Response Evaluator is invoked with the transformation deemed best so far and a list of exact matches, if any, produced.

The Similarity Calculator invokes seven small predicates that test two c-terms for each of the possible similarity metrics. If a similarity predicate is true, the similarity is collected into a similarity set which is subsequently returned by the calculator and assigned to the first c-term. This similarity set can then be ranked according to where it lies in the ordered similarity sets, and this ranking becomes one part of the heuristic required by the Conceptual Distance Calculator.

The Operation Cost Calculator provides the other part of the heuristic measurement required by the Conceptual Distance Calculator. It scans the path of a given transformation of the user's response, noting each operation used to produce each transformation and sums the cost of each, as determined by the plausibility of the inference each makes.

The C-term Transformer is the controlling module for all applications of the transformation rules. It invokes each transformation rule on the current c-term and associates the operation used with the transformation produced. Each transformation rule will produce a complete set of single-step transformations of the current c-term as made possible by the DB and KB. These are single-step in that no more than a single application of the transformation rule in question has produced it. This set of transformations is then sent back to the Conceptual Distance Calculator for insertion into the heuristically ordered set of unexplored c-terms.

The Response Evaluator receives, from the Conceptual Distance Calculator, the path produced by the transformation process and all exact matches found. If none were found, the transformation that came the closest to the expectation is provided. From this information an evaluation category is decided on and, in the current system, this evaluation is output along with pertinent transformations and their paths. In a complete version of MIDMAN, this evaluation will be sent
to the Conversational Move Generator and used to help decide the next conversational move the system should make in its dialogue with the user.

### 6 System Flow

The system first poses its query and accepts a natural language response from the user. This response is then sent to the parser which returns an intermediate representation of the natural language input. This representation is then translated into an $L_k$ expression and given to the module that computes the conceptual distance between the user's response and the system's expectation.

In the distance computation module, the response is first assigned its similarity set in relation to the system's expectation and its heuristic ranking is computed. This ranking is the level of the similarity set assigned plus the operational cost incurred so far (which will be zero at the beginning). The response is then pushed onto the OPEN list. The OPEN list is comprised of c-terms that have yet to be transformed. The search then begins as this c-term is then taken off the OPEN list and pushed onto the CLOSED list. The CLOSED list is a list of all c-terms that have been transformed and is used to prevent cycles in the search. A count of transformed c-terms is kept by incrementing the counter $\text{TRANSMADE}$ at this point. This will be used to limit the number of transformations the system will be allowed to make in its search for ambiguity.

At this point, the similarity set of the response is compared with the best similarity set. If they are the same, the system's expectation has been fulfilled and an exact match has occurred. When an exact match occurs, the system must know if this is the first or a subsequent exact match. If it is the first, an ambiguity check must be initiated; if it is a subsequent exact match, ambiguity has been found. In the first case, the c-term is pushed onto the EXACTMATCHES list and processing is continued with the difference that the $\text{TRANSMADE}$ counter is no longer incremented but decremented as an ambiguity check is pursued. In the second case, the c-term is also pushed onto the EXACTMATCHES list but processing stops and the evaluator is invoked with these multiple exact matches.

If an ambiguity check has been initiated, processing will continue until the $\text{TRANSMADE}$ counter is less than zero or another exact match is found that has the same heuristic ranking as the first. If no other exact match is found within the allotted number of transformations, processing halts and the evaluator is invoked with the single exact match. If another exact match is found that does not exceed the first in heuristic ranking, processing is halted and the evaluator is invoked with both exact matches.

If no exact match has occurred, then two other stopping conditions are checked. If the ranking of the c-term due for transformation exceeds the maximum ranking allowed, or if the number of untransformed c-terms on the OPEN list exceeds the maximum allowed, processing is halted. In both cases, the evaluator is invoked with the c-term deemed BEST-SO-FAR.

The BEST-SO-FAR c-term is the c-term that has the best heuristic ranking as determined by the system to this point in its processing. The BEST-SO-FAR c-term is needed in the case where no exact match is found by the system and it must determine how close the best transformation that could be made from the response is from its expectation. If it is determined that the new transformation has a better heuristic ranking than the c-term considered BEST-SO-FAR then BEST-SO-FAR is assigned the value of the new transformation.

The first actual step of transformation is a call to the Database Consultation module. Here the database is searched for a table that pertains to the c-term at hand. All concept labels and their associated values are gathered from the c-term. If at least one value is unknown and there is at least one constraining value, a table that contains matching attributes for all gathered concept labels is searched for. If such a table is found, a QUEL query is constructed that will elicit the required tuple(s) from it. When this information is returned from the database, it is translated into an $L_k$ expression.

The c-term is then sent to the Transformation module. The Transformation module invokes each of the transformation rules with the current c-term and waits for each to return its set of transformations. Each of the transformation rules receives this c-term and attempts to make all possible single step transformations on it based on knowledge in the KB. When all sets of transformations have been returned to the Transformation module, they are pruned of undesirable c-terms including those that could cause cycles in the search.

At this point in the processing each surviving transformation is assessed for its similarity with the system's expectation and assigned a similarity set. Again this is combined with the operation cost incurred so far to provide a heuristic ranking and each is pushed onto the OPEN list which is reordered by this heuristic ranking, forcing the most promising transformation to the top. The most promising c-term is the one that is the closest to the system's expectation in terms of high similarity and low incurred operation cost. This most promising transformation is then checked for an exact match with
the expectation. If such a match is found and no ambiguity is found, the process halts and the transformation is sent to the Evaluation module to evaluate the quality of the user's response with respect to the system's expectation. If no such match is found the BEST-SO-FAR c-term is checked and the process continues as before, with each transformation itself being transformed until either 1) a exact match is found, 2) no more transformations can be made, or 3) any of the assigned maximums are reached. In the first case the Evaluation module is invoked with this exact match. In the second and third case, since no exact match was ever produced, the transformation deemed BEST-SO-FAR is given to the Evaluation module and a determination of the quality of the user's response is made from it. In any case, the Evaluation module's determination is then output along with the path produced by the conceptual distance calculator.

An example is provided (Figure 2) that provides the path produced for the system query *Where did you do your thesis?* with the user response *My advisor was John Doe.* Each link in the path is annotated with the transformation rule used to produce the next c-term and the knowledge required to make the transformation. The system expectation is UNIVERSITY[val:*TARGET*] and the c-term representation of the user's response is ADVISOR[val:JOHN-DOE]. This path was produced by the system with only four extraneous node expansions; all other Indirect Responses tested yielded paths with fewer extraneous node expansions.

7 Current status

A prototype of MIDMAN has been implemented on a Symbolics Lisp Machine and interfaced with INGRES running on a Sun workstation. The Conceptual Distance Calculator, Response Evaluator, and C-Term Transformer with all the transformation rules have been implemented and tested with a small KB and DB. A number of query-expectation pairs have been devised along with several possible user responses, with at least one of each evaluation category, to these queries. In each case, where an exact match was discovered, either directly or indirectly, the paths that were produced were all optimal. Further, the heuristic focussed the attention of the system so well that in producing this optimal path in the Indirect Response cases on average only took 2.4 more node expansions than required by a direct traversal of the optimal path itself.

Test results show that the notion of Conceptual Distance, when embedded in an appropriate search mechanism, helps produce an optimal path when such exists between two c-terms and provides the linkage necessary
to integrate a database and a knowledge base. This path, when it is between the system's expectation of what form a user's response will be to a given query and the actual response supplied, can subsequently be used to help quantify this response in relation to this expectation and provide guidance for a dialogue manager's next conversational move. This allows a far greater latitude in a relational database interface than previously possible and is a significant step towards mixed initiative two way man-machine communication.

The limited domain of a typical expert system is naturally suited to the approach outlined in this paper. The task remains to structure domain knowledge in such a manner that it is usable by both an inference engine and the dialogue manager. Given that this is possible, the use of such an approach could substantially improve all phases of expert system user interfacing, including knowledge acquisition and elicitation, by allowing the system more flexibility in its dealings with both domain experts and end users alike.

References


