Natural Language Interaction Using a Scalable Reference Dictionary

Veera Boonjing* and Cheng Hsu**

*Mathematics and Computer Science
King Mongkut's Institute of Technology Ladkrabang
Ladkrabang, Bangkok 10520 Thailand
kbveera@kmitl.ac.th
** Decision Sciences and Engineering Systems
Rensselaer Polytechnic Institute
Troy, NY 121800 USA
hsuc@rpi.edu

Abstract: A truly natural language interface needs to be feasible for actual implementation. We developed such a new approach for database query and tested it successfully in a laboratory environment. The new result is based on metadata search, where the metadata grow in largely linear manner and the search is linguistics-free (allowing for grammatically incorrect and incomplete input). A new class of reference dictionary integrates four types of enterprise metadata: enterprise information models, database values, user-words, and query cases using an ontology-based meta-structure. The layered information models allow user-words to stay in original forms as users articulated them, as opposed to relying on permutations of individual words contained in the natural input. These properties make the approach *scalable* to the number of users and the size of the database. A graphical representation method turns the dictionary into searchable graphs representing all possible interpretations of the input. A branch-and-bound algorithm then identifies optimal interpretations, which lead to SQL implementation of the original queries. Query cases enhance the metadata and the search of metadata, as well as provide case-based reasoning to directly answer the queries. This design assures *feasible solutions* at the termination of the search - i.e., the results always contain the correct answer.

1. The Problem of Natural Language Database Query

A truly natural language interface for database query has many important promises; but they all predicate on its being sufficiently ***feasible*** (either in the naturalness or in the scalability) for actual applications. This feasibility is still a hard problem in the field. To illustrate natural language database query, we list below a number of actual queries from mocked end users against a laboratory Computer-Integrated Manufacturing (CIM) database that we tested in the research. Many of them were incorrect or incomplete for the English language, but could happen easily in actuality.

- Just give me only customers who placed orders on PZ1.
- Get billing address of John Smith.
- PZ1 customers
- How about PZ1 orders?
- Give all order_id and models that John Smith's orders.
- Do we have records about John Smith, Jim Kowalski, and William Batt?; they are our clients.
- Not done orders
- Now I want to know cost of PZ1 and PZ10
- What models did John Smith order?
- How about part PZ1?
- All customers.

In these truly natural queries, not only multiple different interpretations exist for the same expressions, the interpretations might not even contain sufficient data to complete the query for processing on the database. Thus, the two basic problems facing natural language database query are ***ambiguities in intent and specification***, and both require the system to possess deep and broad metadata about the underlying database to resolve.

A significant portion of previous results in the field develops specific concept-form-syntax models either for the users to use explicitly (imposed templates) or for the applications to target implicitly (presumed patterns). These models provide precise interpretation and mapping to traditional database query languages when the user input matches exactly. Thus, they control the ambiguities at the expense of naturalness. We might classify these results into five categories according to the technical nature of their linguistic models. They include (1) the template-based approach [e.g., SH01], (2) the syntax-based approach [e.g., AG97], (3) the semantics-grammar-based approach [e.g., COD78; HE78; OW00; WC78], (4) the intermediate-representation-language-based approach [e.g., BAT86; GA87; WU92], and (5) the concept-based models such as MindNet [RI98], WordNet [FE98], and conceptual dependency [SC75]. Some recent works combine these results [MET02] and have also tried to offer more naturalness by narrowing the application domain and engaging more natural language forms and rules [e.g., RO00; SH01].

Another fundamental approach developed previously is to use a general linguistics-string model to allow for interpretation of free-text queries. A prevailing idea of such models is the n-gram-based dictionary, containing all possible uses (permutations) of words found in all past queries to interpret new queries. It then maps the interpretation to a semantic representation of the database structure and thereby determines the executable queries for the original natural language input. Examples include many approaches using relation, object, tree, space vector, probabilistic measures, semantic parsing, and other designs to relate concepts to strings [JK01; JO95; KI00; ME99; MIT02; WA84; ZH99]. Some other results limit the design to particular applications in order to control the size of the models [e.g., BE99; JA86; GT82; ME99; MOT90; SI92]. A concern for these results is their scalability. When the linguistics used are simplistic, such as containing only synonyms to the database objects, the question of multiple interpretations - i.e., the ambiguity in intent

- looms large. When, on the other hand, the models are theoretically comprehensive enough to deal with ambiguities, they may become too costly to implement for non-trivial applications. The size of an n-gram dictionary could increase exponentially as the number of queries and users increases.

In all these approaches, incomplete input could still cause ambiguities in specification and/or intent, if the systems do not possess sufficient metadata to "complete the picture" for the users. Furthermore, a truly natural language interface can never guarantee complete success for all uses at all times - even humans routinely misunderstand other humans. Thus, it has to be able to always yield a feasible solution that contains the correct answer plus some additional but relevant information, and to continuously improve its performance through experience: obtaining (online) users' responses and/or the cases on the users. It follows that interaction with users is always critical for the system to close the loop and assure its validity. However, to make the interaction meaningful, the system has to possess sufficient metadata in order to provide a useful reference scheme on which to base the interaction (e.g., dialogue). Previous results have shown the need for sufficient interaction capability to achieve the above goal [COD78; KA84; MIL75; MY76; RU01; WC78].

Therefore, we submit that ***open***, ***deep and scalable metadata*** about both the structure and application of the databases is a key to solving the problem of natural language query. With sufficient metadata, the system could resolve ambiguity in specification and support a reference dictionary capable of resolving ambiguity in intent that grows linearly. Moreover, with them, the system could develop an efficient interaction with the user to assure a feasible closure for query processing as well as to effect continuous improvement. We envision the following metadata: query cases as in case-based reasoning, an enterprise information model expandable to include any number of semantic layers, database objects and values, and open user-words recognized for the information model and database objects and values. We then develop a ***feasible region*** interpretation approach using metadata to achieve natural language database query, as described in Section 2 below. We present the core method of the new approach next in Section 3, the laboratory testing in Section 4, and discuss some proposed new research afterwards in the concluding section.

2. The Metadata Search Approach

The four types of ***metadata*****: *query cases, enterprise information model, database objects and values, and user-words***, mutually support each other in an integrative manner. Database objects (structure, operators, etc.) and values are what the database is; but users have to refer to them in one way or another in their queries. Enterprise information models represent well-defined applications and the basic concepts of these applications for the underlying databases. An information model could be flat, representing only the immediate database structure and hence becomes fundamentally not scalable; but it could also include scalable layers of semantics on top of the database structure. The more scalable and layered, the more completely this subset of metadata captures the concepts

users use to make queries. Users would, however, use their own words to refer to these concepts and database objects and values when they articulate a request against the database. The individual phrases and words user use in their original forms (not permutations of the original words that the system derives and generates) are the user-words. Query cases are past examples of user queries and their correct results. They should include minimally the exceptions - i.e., the cases where the system would have to engage an interaction with the users to obtain the correct answers. Thus, a case could include user, application and other pertinent information to assist its future use in interpreting the queries from perhaps the same user, the same application and the like. In this sense, cases close the gap between the metadata collected and the ones required at any given point of time. The system could use cases to interpret natural queries directly, if necessary, in a case-based reasoning manner. Clearly, database values, user-words and cases would grow as the use of the database continues; but the growth is proportional linearly to the actual queries processed. Other types of metadata could grow, too, to a much less extent. The scalable enterprise information model is at the core of the metadata. It allows the system to infer the missing information to complete the incomplete queries; and it minimizes the need for user-words. Together with cases, they make it possible for a sufficient reference dictionary to avoid using n-grams. All together, they promise a feasible solution to resolving the ambiguities in intent and specificity.

A natural query is reducible to the set of user-words, concepts, database objects and values it contains. Conversely, a particular combination of these keywords could represent a particular natural query. However, the set of keywords may not always correspond to a unique and complete database query that the system can process correctly. The reasons include possible multiple mappings of the keywords to database objects/values, and incomplete keywords that the systems can complete in multiple ways. This is where the ***search*** approach comes into play. We first store the metadata into a reference dictionary and manage it as a Metadatabase (see the next section), and then use networked graphs to represent their ***logical structure***. An interpretation of a set of keywords is an image of the set on the logical structure. The approach identifies implicitly all permissible interpretations (including the ones derived from the reference dictionary for incomplete input) for the set from the dictionary, and evaluates their relative merits to optimize the interpretation for the query using the branch-and-bound method. The optimal interpretation then leads to a query using the underlying database query language (such as SQL). The above approach is free of the need to understand linguistically the human meaning of the words and phrases used. It needs only one keyword to start the search process, since it could infer possible (multiple) interpretations from the graphs of the logical structure that contain it. When the original user query contains a complete set of metadata free of ambiguity, such as the ones that satisfies an SQL programmer, no derivation is necessary and a solution is readily available. Among all possible interpretations for a set of keywords, the one that involves least addition of derived keywords (including zero) should be the one most accurate to the user's intent. The argument here is simple: users would not be deliberately wordy with respect to their own standards, nor misleading, when querying a database. Therefore, minimal sufficiency is a good search criterion while the logical structure is the constraint of the metadata search algorithms, for a constrained optimization paradigm.

The core of the reference dictionary (enterprise information model and initial user-words) will be a design-time product; while cases, additional user-words, changes to the information model, and database values will be added as the database system evolves. The reference dictionary-based logical structure holds four fundamental promises: it generates search-ready graphics; it supports case-based reasoning; it assures complete interpretations of natural queries; and it simplifies user-words. The last point is worth further elaboration. Consider the theoretical complexity of processing only one natural query. Suppose the text consists of a stream of **n** words of which **m** are recognized metadata. There would then be **n/m** words associated with each known term; these words become the candidate user-words for the known terms. The expected number of permutations of these new words would be **m*(n/m)!** for the query. As **n** grows over time with new queries, the number of user-words would grow in this rate - and this is the general complexity of a linguistic dictionary. Obviously, the system wants to increase the number **m** (hits) since the bigger **m** is, the fewer (exponentially) the possible groupings of words would be, resulting in fewer new user-words considered or added to the dictionary. Information models with rich (layered) semantics provide a large **m** for the initial design of user-words, and consequently lead to less ambiguity, fewer possible interpretations, and fewer new user-words. When **m** reaches a reasonable scale, permutations of user-words become less necessary. Cases do not directly change **m**, but do help resolve ambiguity due to insufficient user-words, and hence help to reduce the need for new user-words. Information models and cases represent a tightly structured, efficient kernel of meaning with which the users will be familiar.

The above argument shows a strong probability that the new method is scalable to the number of users (queries) and the size of the database, since the reference dictionary grows linearly. This scalability gets stronger when the meta-structure of the reference dictionary itself allows for open and scalable management (addition, deletion, and modification) of enterprise metadata as the underlying database systems evolve. This indeed the case as we will discuss in the next section.

The core logic of the metadata search approach proceeds this way. Given a graph **R = <V, E>** of the reference dictionary where **V** is a set of vertices consisting of cases (**C**), user-words (**U**), information models (**M**), and database values (**D**); and **E** is a set of edges connecting them:

Step 1: Identify an ordered k-tuple $\mathbf{I}_{keyword} = (t_1, t_2, \ldots, t_k)$ where t_i is a word/phrase (keyword) in the input that also belongs to **U**, **M** or **D** of **R**; i = 1, 2, …,k; and k is a number of keywords found in the input. Associated with each element t_i is the set of referenced **M** or **D** (each element of $\mathbf{I}_{keyword}$ may refer to many elements in **M** or **D**). Denote this set as $\mathbf{V}_i$. Therefore, we have an ordered k-tuple $\mathbf{V}_{keyword} = (V_1, V_2, \ldots, V_k)$ where V_i is a set of referenced **M** or **D** for t_i.
Step 2: Determine the most similar past case by matching keywords of the query with problem definitions of past cases. If a perfectly matched case is found, then apply the case solution and go to Step 5. Otherwise, if the similarity measure of the most similar case is sufficiently significant (say at least 60%), then modify the case to obtain a solution and go to Step 5.
Step 3: Determine a minimal combination set of elements of $V_1, V_2, \ldots, V_k$.

Step 4: Search for the best interpretation by using the branch and bound method.
Step 5: Map the result to the database query language. Obtain the results of the query and confirm them with users.

Note that the case-based learning mechanism engages user in dialogue as needed. Its outcome becomes new cases and user-words added to **C** and **U**, respectively. The above logic consists of a few postulates.

Postulate 1: Necessary Condition
The natural query input contains at least one keyword found in **R**.
Postulate 2: Sufficient Condition
The natural query input contains an unambiguous set of keywords necessary for forming an accurate SQL statement for the query.
Postulate 3: Optimality
When all other things are equal, the best interpretation of the natural query input is the one requiring minimum traversal on the logical structure of **R**.

Although the sufficient condition guarantees the logic to succeed, the approach actually works under the necessary condition in most cases in our testing. Postulate 3 also allows for adding new criteria such as operating rules for the database to the search.

3. The Core Methods and Algorithms: An Illustrative Example

The above basic logic is implemented in a set of core methods and algorithms, including an ontology-based meta-structure of the reference dictionary, a graphical representation of the metadata in the reference dictionary, a branch-and-bound search algorithm, and a case-reasoning logic. The details are available from the literature [BO02]. We give a conceptual overview here. The meta-structure is based on the Metadatabase model [HU96], using the Two-Stage Entity-Relationship ontology of information modeling [HU96]. The Metadatabase is shown open and scalable to incorporating heterogeneous databases into the integration environment. Thus, the reference dictionary is capable of integrating multiple information models and accommodating their evolution online. This research adds some extensions, namely cases and user words, to the previous model. A case in the case-based reasoning paradigm typically consists of three components: a problem definition, a solution, and its outcome. In this research, the reference dictionary contains the complete domain knowledge needed, so the problem definition is expanded but the outcome is dropped. New problems would use the problem definition to find the (best) matching cases and apply the associated solutions to them. A set of keywords and their recognized vertices for a query describes the problem definition, and its interpretation defines the solution.

The graphical representation defines the logical structure of a database on a graph G, a sub-graph of the reference dictionary graph. Given a *natural language query* Q, the natural language interface performs an interpretation in several steps. It first scans Q to recognize the keywords (entries in the reference dictionary) in the natural query, i.e., the

recognized keyword. It then determines their corresponding *recognized vertex sets* in G and identifies all *query images* of Q on G based on these vertex sets. Since Q may be ambiguous (e.g., incomplete and multi-valued mapping) to G, each of its recognized keywords could correspond to multiple recognized vertices, resulting in multiple query images. Further, a recognized vertex may not always connect to other recognized vertices in a way covering a complete range of data semantics (database value, attribute, entity, and relationship) with a unique path. Therefore, it could have multiple *semantic paths*. Taking these data semantics into account, we obtain all possible semantic paths for a query image, called the *feasible graphs*. The refinement of feasible graphs leads to *connected feasible graphs* and *complete query graphs*. *A complete query graph represents an executable interpretation of the natural language query Q according to the logical structure G*. The optimization algorithm implicitly searches all possible interpretations to determine the final query graph for execution.

The graphical representation method yields a feasible region poised for search. As long as the search method employed is bound within the region, then any results obtained are guaranteed to be a feasible solution relevant to the original query. The solution may not offer pinpoint precision, i.e., may contain additional information not requested in the original query, but will nonetheless include the correct answer and the additional information will be meaningful. A case would be a query to get the cost of part ID 111222 that returned an answer of all information for the part including the cost. The answer is clearly feasible and useful to the user.

The search algorithm itself follows the Branch-and-Bound logic, as illustrated below in the processing of the first natural language query example presented in Section 1: Just give me only customers who placed orders on PZ1. Henceforth this query is referred to as Query 1.

Suppose keywords (words and phrases matching some entries in the reference dictionary or some vertices in graph G) and their recognized vertices (a set of G vertices matching keywords) for this query are as shown in Table 1.

Table 1. keywords and their minimal vertex sets for Query 1.

Keyword	Vertex
CUSTOMERS	S CUSTOMER
ORDERS	S ORDER
PZ1	V opsI_100|PZ1, V ppsI_54|PZ1, V sfcI_11|PZ1, V sfcI_5|PZ1

The followings are possible query images generated from Table 1.

QI 1: { S CUSTOMER, S ORDER, V opsI_100|PZ1 }
QI 2: { S CUSTOMER, S ORDER, V ppsI_54|PZ1 }
QI 3: { S CUSTOMER, S ORDER, V sfcI_11|PZ1 }
QI 4: { S CUSTOMER, S ORDER, V sfcI_5|PZ1 }

The search graph of branch-and-bound search for Query 1 is shown in Figure 1. It branches from root vertex where its lower bound (LB) = 1 to all possible query images (QI1-QI4). See [BO02] for details on the evaluation function LB(). Among these query images, the QI1 has the least lower bound. Therefore, the next vertex to explore is QI1.

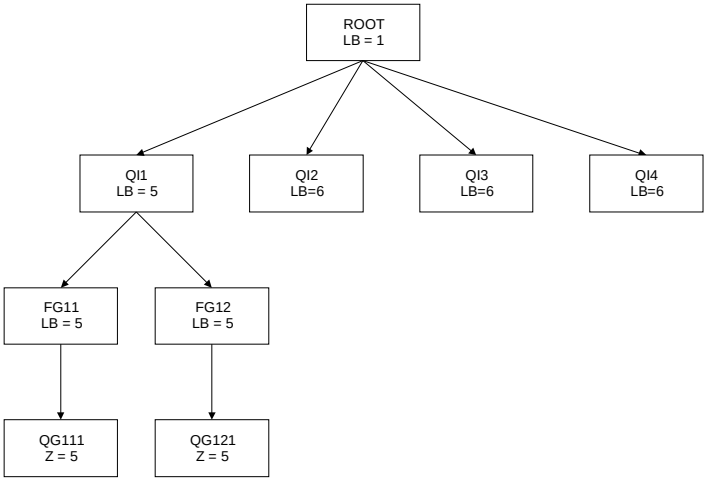

Figure 1: The search graph for Query 1

There are two possible feasible graphs can be inferred from QI1. Both FG11 and FG12 have LB = 5. Since this lower bound is still lower than of any other vertices visited, the branch-and-bound search branches from FG11 and FG12. The branch from FG11 give the query graph QG111 with cost = 5 as shown in Figure 2. The branch from FG12 also gives the query graph QG121 with cost = 5 as shown in Figure 3. No other vertices visited can give a cost lower than this. Therefore, both QG111 and QG121 are considered as interpretation candidates for Query 1.

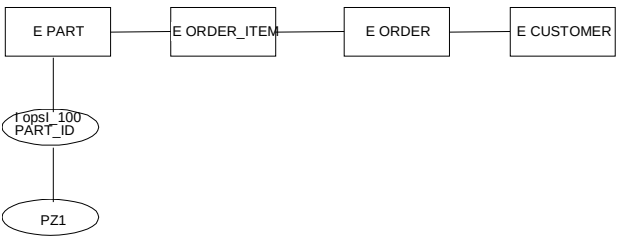

Figure 2: The query graph QG111 for FG11

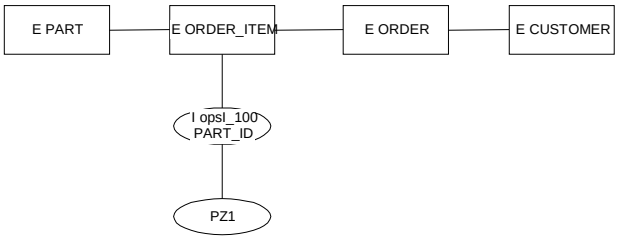

Figure 3: The query graph QG121 for FG12

Consider QG111 and QG121, both of them contain the same attribute and value and the same entity/relationship solution path. The only difference is the attribute and value belonging to the different entity/relationship in the path. This happens because an attribute could be a key in one entity/relationship and a foreign key in other entities/relationships. Since it does not make any difference in semantics between QG111 and QG121, they are considered equivalent. Therefore, one of them can be removed.

Suppose the QG121 is removed, then the QG111 is the final solution. This interpretation is then mapped to the following SQL query. See [BO02] for mapping rules.

```
SELECT DISTINCT PART.COST, PART.DESCRIPTION, PART.PART_ID,
ORDER_ITEM.ORDER_LINE_ID, ORDER_ITEM.CUST_ORDER_ID,
ORDER_ITEM.DATE_SCHED, ORDER_ITEM.QUANTITY,
ORDER_ITEM.OI_STATUS, ORDER_ITEM.PART_ID,
ORDER.CUST_ORDER_ID, ORDER.DATE_DESIRED,
ORDER.OD_STATUS, ORDER.CUST_ID, CUSTOMER.CUST_ID,
CUSTOMER.CUST_NAME, CUSTOMER.B_ADDR, CUSTOMER.S_ADDR
FROM ORDER, CUSTOMER, ORDER_ITEM, PART
WHERE ORDER.CUST_ID=CUSTOMER.CUST_ID and
ORDER.CUST_ORDER_ID=ORDER_ITEM.CUST_ORDER_ID and
ORDER_ITEM.PART_ID=PART.PART_ID and
(PART.PART_ID = 'PZ1');
```

4. Extension: Case-Based Reasoning and Interaction

In theory, the system could accumulate all natural queries it has processed (using the above method) into a case-base. When the case-base has grown to a sufficient size, the designer would have a choice to shift to relying mainly on case-based reasoning to answer new queries and uses the interpretation algorithm only as a backup. This design would be reasonable since the growth of cases is linear. However, in this research, we still regard initially cases as a secondary tool to support the primary means of metadata search. As such, the role of case-based reasoning is to assure a closure to the user and in a sense a learning capability to the system. Therefore, we consider in this paper only the need to build new cases for natural queries that the system failed to provide a single (query graph) answer or the user rejected the answer. An interaction with the user is the means to

achieve this end. The interaction capability is more a last defense than a regular way of solution; however, it could improve the performance in the worse than average cases when the repository of the cases grows. When an interaction becomes necessary, the system asks the user for additional information to solve the query correctly. There are three possible causes of failure: (1) incorrect recognized vertex for a recognized keyword, (2) no recognizable keywords in a query, and (3) insufficient keywords. A metadata-based interaction identifies the cause and guides the user to define new user-words to fix it. For the first cause, it provides the pertinent metadata (information model) describing the possible recognized vertices for the keywords found so that the user could either designate the intended vertex for a keyword or redefine the keywords. For the second cause, it walks the user through the enterprise information model from high level down to associate new user-words to vertices of information models. For the last cause, it similarly guides the user to provide additional keywords for the query. In all these cases, the interaction adds new user-words as well as building cases to the reference dictionary. The system will repeat this cycle until it yields a correct result. Another design to resolve the case of multiple query graphs for a query is to present all possible solutions to users in terms of natural language for them to choose the intended solution. If the number of alternative solutions is relatively large, it presents all possible scopes of solutions for users to choose to narrow down the possible solutions before asking them to choose the intended solution. When the system obtains the correct result, it completes the case with problem definition, solution definition, and other information according to the structure of cases in the reference dictionary. The method of case-based reasoning uses the cases built and accumulated either to assist the metadata search or to answer directly the queries. The method includes (1) a matching mechanism to match a query and a case, (2) criteria to decide whether to reuse the most similar case, and (3) a mechanism to reuse the case. We adopt standard results in the information retrieval field, with modifications to accommodate equivalent keywords, to match a query with a case, viz., the vector space model [SA83] and its COSINE measure of similarity. The actual design will be a major task of the proposed research.

The above extensions also add to the basic scalability of the new method, in the sense that they facilitate the performance in actual implementation. We verify the claims about scalability through a laboratory test, as discussed next.

5. Verification at a Laboratory

We implemented the core algorithms in a software system running on a UNIX-Oracle Server. The reference dictionary and the application database - a Computer-Integrated Manufacturing (CIM) system - both reside in Oracle. The software itself has two components: the user interface, coded in HTML and Java Script to be compatible with major Web browsers, and the core algorithms, using Java Applets embedded in the user interface in a browser-based computing manner. The implementation developed so far solves primarily natural language queries that have a basic SQL solution. That is, the basic SQL uses search-conditions in the WHERE clause that either (1) are an expression of the form "$attribute_1$ = $attribute_2$" or "attribute = value," or (2) a combination of such

expressions with the logical operators “AND” and “OR.” These results are sufficient to show the feasibility of the metadata search approach but are insufficient for implementing the approach in practice.

We tested the prototype system with 10 users to substantiate the feasibility of the metadata search approach. The participants included professionals not associated with Rensselaer as well as undergraduate students and graduate students at Rensselaer. Their background in databases ranges from strong to virtually none. We provided the participants with the descriptions, information models, and contents of the CIM database; but were otherwise completely detached from them so as to allow them to ask as many questions as they wished against the database in any text forms. An answer to a question is correct when it includes all information the user required. In this implementation, the reference dictionary of the testing case included only information models, database values, and less than 200 user-words that we created by ourselves in advance.

The system solved every query listed in the first section, The Problem. It used a limited number of new user-words to solve every additional query listed below. It failed originally to produce correct answers to these queries; however, it solved every one when the boldface words/phrases that the users articulated were added to the reference dictionary as user-words (only one entry for each word or phrase). Without them, the system produced either multiple interpretations for the user to select or a single incomplete interpretation. Either cases contained the correct interpretation. There were no other queries users generated that the system failed to solve correctly.

- William Batt's **contact**
- Michael Goin **address**
- Not done yet **transaction**
- **Buyers** of PZ10.
- How many **people** order pz1?
- Is Jim Kowlaski billing address and **mailing address** the same?
- What is Craig Norman billing address and **home address**? Please also list all of his orders.
- What are the **products** associated with order no 00001?
- Show the detail Craig Norman’s order, part description, quantity, and **where to ship** it?
- List all **in-process** orders and their details.

The important fact about these new user-words is their consistency with the basic logic of metadata search: *they do not require n-gram and their number grows linearly*. The result shows, the concept of metadata search is sound and the approach is scalable. It also supports the claim that enterprise information models with layered semantics reduce the number of user-words required and enhance the metadata search.

All queries tested satisfy the necessary condition and the sufficient condition postulated in Section 2. The response times of all queries were between 1 and 10 seconds, running the software from a Web client site against the Oracle server.

6. Beyond the Current Results

Many scholars consider natural language interaction a major pillar of Information Technology in the new century. However, the field has not been as active as it should be, because in part of past frustrations. The research reported here contributes to enterprise database systems, which have good applications in the field of Internet-based information services. In a broader sense, the work might show that natural language processing is actually feasible, in the sense that it almost always produces a useful answer (feasible region-bound) to almost any relevant query, and that companies can use it at reasonable cost. The preliminary results suggest that natural language processing could apply to a significant domain where many enterprise users need completely intuitive user interfaces (allowing for ambiguous, incomplete, and incorrect articulation) to fully reap the benefits of online enterprise information. The continued research on this line promises to show the viability of having end users querying databases in their own languages. The medium of query could either be free text typed in the system or be natural sentences spoken to it.

The continuing research will first extend the core method to include the full range of SQL capabilities, especially the other standard operators not included yet, derived-attributes and expressions, and date-related and other temporal selection criteria. We need to develop a large-scale case-base to provide a parallel approach for query interpretation. Second, research on implementation tools and methodology to help acquire the metadata and maintain the reference dictionary is required. Third, opportunities exist in connecting the system to voice interaction using even commercially available voice technologies to allow for voice query of databases. They also exist in the exploration for applications, especially in the Web-based settings. Finally, the possibilities of creating sufficient metadata to represent the application domain of information retrieval outside of databases deserve exploration. Research in this area would engage directly the progress on the general problem of human computer interaction.

References

[AG97] Adam, N.R. and A. Gangopadhyay, "A Form-Based Natural Language Front-End to a CIM database," *IEEE Trans. On Knowledge and Data Engineering*, 9:2, 1997, pp. 238-250.

[BAT86] Bates, M., M.G.Moser, and D. Stallard, "The IRUS Transportation Natural Language Interface," in *Expert Database Systems*, 1986, pp. 617-630.

[BE99] Bergman, L., J. Shoudt, V. Castelli, and C. Li, “An Interface Framework for Digital Library,” *International Journal on Digital Libraries*, 2:2-3, 1999, pp. 178-189.

[BO02] Boonjing, V. and C. Hsu, “Metadata Search: A New Approach to Natural Language Database Interfaces,” *Proceedings IASTED International Conference on Information Systems and Databases*, October, 2002, Tokyo, Japan.

[COD78] Codd, E.F., "How about Recently? (English Dialog with Relational Databases Using Rendezvous Vesion 1). In B. Shneiderman (Eds). Databases: Improving Usability and Responsiveness, 1978, pp. 3-28.

[FE98] Fellbaum, C. (Eds), “WordNet: an Electronic Lexical Database,” MIT Press, Boston, 1998.

[GA87] Gross, B.J., D.E. Appelt, P.A. Martin and F.C.N. Pereira, "TEAM: an Experiment in Design of Transportable Natural-Language Interfaces," *ACM Trans*actions, 32, 1987, pp. 173-243.

[GT82] Guida, G. and Tasso C., "NLI: A Robust Interface for Natural Language Person-Machine Communication," *Int. J. Man-Machine Studies*, 17, 1982, pp.417-433.

[HE78] Hendrix, G.G. Sacerdoti, E.D. Sagalowicz, C. and Slocum, J., "Development a Natural Language Interface to Complex Data," *ACM Trans. on Database Systems*, 3:2, 1978, pp. 105-147.

[HS91] Hsu, C., M. Bouziane, L. Ratter, and L. Yee, "Information Resources Management in Heterogeneous, Distributed Environments: A Metadatabase Approach," *IEEE Trans. on Software Engineering*, 17:6, 1991, pp. 604-625.

[HU96] Hsu, C., *Enterprise Integration and Modeling: the Metadatabase Approach*, Kluwer Academic Publishers, Boston, 1996.

[JA86] Janus, J.M., "The Semantics-based Natural Language Interface to Relational Databases," in L. Bolc and M. Jarke (Eds), *Cooperative Interfaces to Information Systems*, pp. 143-187, Springer-Verlag, New York, 1986.

[JK01] Jarvelin, K., J. Kekalainen and T. Niemi, "Concept-Based Query Expansion and Construction," *Information Retrieval*, 4:3/4, 2001, pp. 231-255.

[JO95] Johnson, J.A.,"Semantic Relatedness," *Computers Math. Applic*, 29:5, 1995, pp. 51-63.

[KA84] Kaplan, S.J., "Designing a Portable Natural Language Database Query System," *ACM Trans. on Database Systems*, 9:1, 1984, pp. 1-19.

[KI00] Kim, W., "Corpus-Based Statistical Screening for Phrase Identification," *Journal of the American Medical Information Association*, 7:5, 2000, pp. 499-511.

[ME99] Meng, F. and W. Chu, "Database Query Formulation from Natural Language using Semantic Modeling and Statistical Keyword Meaning Disambiguation," Technical Report CSD-TR 99003, 1999, Computer Science Department, University of California, Los Angeles, California.

[MET02] Metais, E., "Enhancing Information Systems Management with Natural Language Processing Techniques," *Data and Knowledge Engineering*, 41:2-3, 2002, pp. 247-272.

[MIL75] Miller, P.L., "An Adaptive Natural Language System that Listens, Asks, and Learns," in *Advance Papers of Forth International Joint Conference on Artificial Intelligence*. pp. 406-413, Morgan Kaufmann, California, 1975.

[MIT02] Mittendorfer, M. and W. Winiwarter, "Exploiting Syntactic Abalisys of Queries for Information Retrieval," *Data and Knowledge Engineering*, 42:3, 2002, pp. 315-325.

[MOT90] Motro, A., "FLEX: A Tolerant and Cooperative User Interface to Databases," *IEEE Transactions on Knowledge and Data Engineering*, 2:2, 1990, pp. 231-246.

[MY76] Mylopoulos, J. Borgida, A. Cohen, P. Roussopoulos, N. Tsotsos, Jr. and H. Wong, "TORUS: A Step Towards Bridging the Gap between Data Bases and the Casual User," *Information Systems*, 2:1, 1976, pp.49-64.

[OW00] Owei, V., "Natural Language Querying of Databases: an Information Extraction Approach in the Conceptual Query Language," *Int. J. Man-Machine Studies*, 53, 2000, pp.439-492.

[RI98] Richardson, S.D., W.B. Dolan and L. Vanderwende, "MindNet: Acquiring and Structuring Semantic Information from Text," In *ACL'98 CONF 17*, 2, 1998, pp. 1098-1102.

[RO00] Rosenfeld, R., X. Zhu, S. Shriver, A. Toth, K. Lenzo and A. W. Black, "Towards a Universal Speech Interface," In *Proc. ICSLP 2000*, 2000.

[RU01] Ruber, P., "Asking Naturally," *Knowledge Management*, 4:8, 2001, pp. 62-63.

[SA83] Salton, G. and Mcgill, M., "Introduction to Modern Information Retrieval," Mcgraw-Hill, New York, 1983.

[SC75] Schank, R.C., "Conceptual Dependency Theory," In R.C. Schank, *Conceptual Information Processing*, pp. 22-82, American Elsivier, North-Holland, 1975.

[SH01] Shankar, AJ. and Yung, Wing, "gNarLi: A Practical Natural Language Interface," http://www.people.fas.havard.edu/~shankar2/stuff/gnari.html, 2001.

[SI92] Shimazu, H., S. Arita, and Y. Takashima, "CAPIT: Natural Language Interface Design Tool with Keyword Analyzer and Case-based Parser," *NEC Res. & Develop*, 33:4, 1992, pp.679-688.

[WA84] Wald, J.A. and Sorenson, “Resolving the Query Inference Problem using Steiner Trees,” *ACM Transactions on Database Systems*, 9:3, 1984, pp. 348-368.

[WC78] Waltz, D.L., "An English Language Question Answering System for a Large Relational Database," *Communications of the ACM*, 21:7, 1978, pp.526-539.

[WU92] Wu, W. and D.M. Dilts, “Integrating Diverse CIM Data Bases: The Role of Natural Language Interface,” *IEEE Trans. Systems, Man, and Cybernetics*, 22:6, 1992, pp.1331-1346.

[ZH99] Zhang, G. Chu, W.W. Meng, F. and Kong G., “Query Formulation from High-level Concepts for Relational Databases,” In N. W. Paton and T. Griffiths (eds), *Prodeedings in User Interfaces to Data Intensive Systems*, pp. 64-74, The IEEE Computer Society, 1999.