Multilingual Question Answering with High Portability on Relational Databases

Hanmin Jung
Department of Computer Science and Engineering
Pohang University of Science and Technology
San 31, Hyoja-dong, Nam-gu, Pohang, Kyungbuk, Korea (790-784)
Telephone: +82-54-279-5581
jhm@postech.ac.kr

Gary Geunbae Lee
Department of Computer Science and Engineering
Pohang University of Science and Technology
San 31, Hyoja-dong, Nam-gu, Pohang, Kyungbuk, Korea (790-784)
Telephone: +82-54-279-5581
gblee@postech.ac.kr

Abstract

This paper describes a highly-portable multilingual question answering system on multiple relational databases. We apply semantic category and pattern-based grammars, into natural language interfaces to relational databases. Lexico-semantic pattern (LSP) and multi-level grammars achieve portability of languages, domains, and DBMSs. The LSP-based linguistic processing does not require deep analysis that sacrifices robustness and flexibility, but can handle delicate natural language questions. To maximize portability, we drive various dependent parts into two tight corners, i.e., language-dependent part into front linguistic analysis, and domain-dependent and database-dependent parts into backend SQL query generation. Experiments with 779 queries generate only constraint-missing errors, which can be easily corrected by adding new terms, of 2.25% for English and 5.67% for Korean.

Introduction

As a natural language (NL) interface, question answering [7] on relational databases [1] allows users to access information stored in databases by requests in natural language [16], and generates as output natural language sentences, tables, and graphical representation. The NL interface can be combined with other interfaces to databases: a formal query language interface directly using SQL, a form-based interface with fields to input query patterns, and a graphical interface using a keyboard and a mouse to access tables. The NL interface does not require the learning of formal query languages, and it easily represents negation and quantification [4], and provides discourse processing [8].

The use of natural language has both advantages and disadvantages. Including general NLP problems such as quantifier scoping, PP attachment, anaphora resolution, and elliptical questions, current NLIDB has many shortcomings [1]: First, as a frequent complaint, it is difficult for users to understand which kinds of questions are actually allowed or not. Second, the user assumes that the system is intelligent; he or she thinks NLIDB has common sense, and can deduce facts. Finally, users do not know whether a failure is caused by linguistic coverage or by conceptual mismatch. Nevertheless, natural language does not need training in any communication media or predefined access patterns.

NLIDB systems [2], one of the first applications of natural language processing, including "LUNAR" were developed from the 1970s [23]. In the 1980s, research focuses on intermediate representation and portability, and attempts to interface with various systems. CHAT-80 [22] transforms an English query into PROLOG representation, and ASK [20] teaches users new words and concepts. From 1990s, commercial systems based on linguistic theories such as GPSG, HPSG, and PATR-II appear [13], and some systems attempt to semi-automatically

[1] We also call it NLIDB (**N**atural **L**anguage **I**nterface to **D**ata**B**ases).

construct domain knowledge. MASQUE/SQL [1] uses a semi-automatic domain editor, and LOQUI [3], a commercial system, adopts GPSG grammar. Meanwhile, Demers introduces a lexicalist approach for natural language to SQL translation [6], and as the CoBase project of UCLA, Meng and Chu combine information retrieval and a natural language interface [14].

The major problems of the previous systems are as follows. First, they do not effectively reflect the vocabulary used in the description of database attributes into linguistic processing. Second, they require users to pose natural language queries at one time using a single sentence rather than give the flexibility by dialog-based query processing. The discordance between attribute vocabulary and linguistic processing vocabulary causes the portability problem of domain knowledge from knowledge acquisition bottleneck; the systems need extensive efforts by some experts who are highly experienced in linguistics as well as in the domain and the task.

Androutsopoulos [1] [2], which are mainly referenced for this section, classifies NLIDB approaches into the following four major categories.

Pattern matching systems: Some of the early systems exclude linguistic processing. They are easy to implement, but have many critical limitations caused by linguistic shallowness [17].

Syntax-based systems: They syntactically analyze user questions, and use grammars that transform parsed trees to SQL queries [23]. However, the mapping rules are difficult and tedious to devise, which drops the portability of languages and domains.

Semantic grammar systems: The systems adopt techniques interleaving syntactic and semantic processing, and generate SQL queries from the result [19] [21]. They are useful to rapidly develop parsers in limited domains, but are not ported well to new domains due to hard-wired and domain-dependent semantic information [18].

Intermediate representation language systems: Most current systems place an intermediate logical query between NL question and SQL [5]. The processes before the intermediate query are defined as the linguistic front-end (LFE), and the other processes as the database back-end (DBE). This architecture has the merits that LFE is DBMS-independent and an inference module can be placed between LFE and DBE. However, the limitation of parsing and semantic analysis requires semantic post-processing. Nevertheless, it is difficult to achieve high quality analysis for database applications.

On the contrary, we apply linguistic processing based on lexico-semantic patterns (LSP), a prominent method verified in text-based question answering [10] [12], into NLIDB, and propose multi-level grammars to represent query structures and to translate into SQL queries. Our system is a hybridization of the pattern matching system and the intermediate representation language system. However, our LSP-based pattern covers lexical to semantic matching, and the multi-level grammars for intermediate representation evidently separate the database back-end from the linguistic front-end. Thus, our method has the ability to divide LFE and DBE, but promises greater adaptability due to the hybrid linguistic analysis and the pattern-matching characteristics.

The LSP-based linguistic processing does not require deep analysis that sacrifices robustness and flexibility, but handles delicate NL questions. To maximize portability of languages, domains, and DBMSs, we drive the various dependent parts into two tight corners, i.e., the language-dependent part into front linguistic analysis, and the domain-dependent and database-dependent parts into backend SQL query generation. In our LSP description, attribute vocabularies are also represented as semantic classes that represent semantic meaning of words. Thus, the domain-dependent attributes and values are automatically extracted from databases, and get registered in a semantic category dictionary.

1 LSP-based Linguistic Processing and Multi-level Grammars

A lexico-semantic pattern (LSP) is the structure where linguistic entries and semantic types can be used in combinations to abstract certain sequences of words in a text [12] [15]. Linguistic entries consist of words, phrases and

part-of-speech tags, such as “television,” “3D Surround,” and “NP[2].” Semantic types consist of attribute names, semantic tags (categories)[3] and user-defined semantic classes[4], such as “@model,” “@person,” and “%each.”

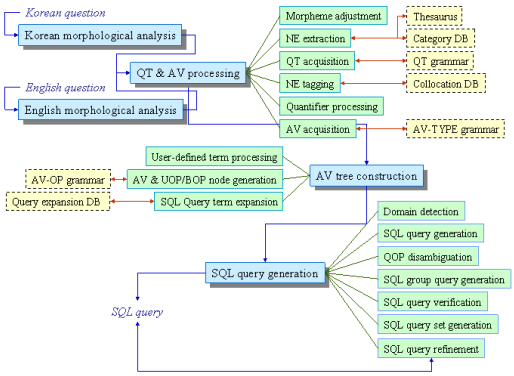
LSP-based language processing simplifies the natural language interface due to the following characteristics: First, linguistic elements from lexicons to semantic categories offer flexibility in representing natural language. Second, simple LSP matching without fragile high-level analyses assures a robust linguistic model. Third, the use of common semantic types among different languages reduces the burden of cross-linguistic portability, i.e., enhances multilingual expansion. Finally, separation between dictionary and rules easily enriches domain knowledge by minimizing the conflict to describe the rules.

Multi-level grammars are designed to construct intermediate representation as the source of SQL query generation. The grammars interpret lexico-semantic patterns obtained from the linguistic front-end, i.e., morphological analysis, and build attribute-value trees for database back-end. We introduce three-level grammars that include lexico-semantic patterns to describe their rules: a QT[5] grammar to determine question types, an AV[6]-TYPE grammar to construct attribute-value nodes (see section 2.1), and an AV-OP grammar to find the relations between the nodes (see section 2.2). Using the QT grammar, query-to-LSP transfer makes a lexico-semantic pattern from a given question [9]. The lexico-semantic patterns enhance information abstraction through many-to-one mapping between questions and a lexico-semantic pattern.

2 System Configuration

To handle the two major problems of previous NLIDB systems, i.e., 1) the discordance between attribute vocabulary and linguistic processing vocabulary, which causes laborious low portability in multiple environments, and 2) the absence of query refinement supporting session-based dialog, our system effectively develops LSP-based linguistic processing, multi-level grammars, and SQL query refinement. To maximize portability of multiple environments, such as languages, domains, and DBMSs, each environment-dependent module is clearly defined and confined.

Our system consists of four phases (figure 2.1): morphological analysis, QT/AV Processing, AV-tree construction, and SQL query generation. The QT/AV processing determines the question types for a given question, and constructs attribute-value nodes from the question. This phase includes linguistic front-end processes, such as morphological analysis and named entity recognition. The AV-tree construction phase finds the relation between the nodes obtained from the previous phase, and produces an attribute-value tree that is independent of DBMSs. The last phase, SQL query generation, translates the attribute-value tree into a DBMS-dependent SQL query.



[Figure 2.1] System Architecture

2.1 Morphological Analysis and QT/AV Processing

We separate all processes and resources using a linguistic dependency approach. Morphological analysis and all the resources, including grammars and dictionaries, are all language dependent. Morphological analysis for each language produces a sequence of (word, lemma, POS) pairs. After the analysis, system shares all

---

[2] Part-of-speech tag for a proper noun

[3] “@” is the start symbol for the semantic tags.

[4] “%” is the start symbol for the user-defined semantic classes.

[5] Question type

[6] Attribute and value

processes independent of the source language; this increases linguistic portability by pushing the language-dependent processes to the earlier stage.

To acquire the named entities for target databases, the system looks up the category dictionary which includes main semantic information. The category dictionary consists of four components: semantic tags, user-defined semantic classes, part-of-speech (POS) tags, and lexical forms. The structure of semantic tags is a flat form. In a lexico-semantic pattern, each semantic tag follows a "@" symbol. For example, a semantic tag "@item[7]" includes the words, such as "비디오," "비데오," and "브이티알" in Korean, and "VCR" and "video" in English. User-defined semantic classes are the tags for syntactically or semantically similar lexical groups. For example, a user-defined semantic class "%each" includes the words, such as "각," "각품목," "개별," and "별" in Korean, and "each" and "every" in English. The category dictionary has the highest priority to construct the lexico-semantic pattern for a sentence. In the absence of an entry, the part-of-speech tag of the current morpheme becomes the component of the LSP.

A question type indicates ordering clauses, including "ASC" and "DESC," or column functions such as "SUM()," "AVG()," and "MIN()." By applying a QT grammar, a question type and a target attribute, i.e., the argument of the question type, are obtained. The following shows the process from user query to SQL template.

[User query]
"How much is the cheapest among 34 inches?"
[LSP pattern]
%how %much %be %dt %most-cheap %among num @unit_length sent
[QT grammar]
……
^[8]***%how%much%bedt%most-cheap***
    qo_min|qt_price
@corpin%asc
    qo_aso|qt_corp
%corp%make@type%and@type%together
    qo_intersect|qt_corp
……
[Question type]
MIN(*price*)
[SQL template]
SELECT MIN(*price*) **FROM** … **WHERE** …

AV-TYPE grammar finds all the pairs of attributes and values by applying lexico-semantic patterns. Like QT grammar, LSP-based condition and action exist. The action consists of an attribute and a value operator. In SQL, the two components are represented like ([attribute] [value operator] [value]) in WHERE clauses, for example, (*price > 300,000*). The pairs of attribute and value become the nodes of the AV-tree, which is the source to generate the SQL query. The following examples demonstrate the method to obtain pairs using the AV-TYPE grammar.

[User query]
"Choose only **Samsung**'s among **25 inches**, **21 inches**, and **29 inches** products."
[LSP pattern]
… @corp … num @unit_length … num @unit_length … num @unit_length …
[AV-TYPE grammar]
……
engj%start
    @model|vo_begin
num@unit_length
    @size|vo_like
%betweennum%andnum
    @price|vo_between
%each%price
    @price|vo_group
@corp
    @corp|vo_like
@price%elower
    @vop_mod[9]|vo_elower
……
[Pairs of attribute and value]
(corp like *'%Samsung%'*)
(size like *'25inch'*)
(size like *'21inch'*)
(size like *'29inch'*)

[7] "item" attribute for "audio-video" product database

[8] Symbol designating the beginning position of a regular expression

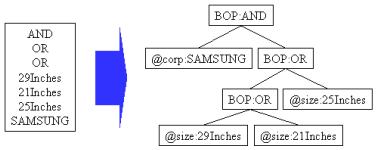
[9] Action to modify the current value operation

2.2 AV-tree Construction

Binary operators (AND, OR, NOR, NAND …) connect the pairs of attributes and values that are the nodes of the AV-tree. AV-OP grammar describes the relations using lexico-semantic patterns like other grammars. The condition part of the grammar consists of attributes and conjunctions (see examples below), whereas the action part consists of binary operators and the attributes' index in postfix notation.

[Pairs of attribute and value from section 2.1]
(corp like *'%Samsung%'*)
(size like *'25inch'*)
(size like *'21inch'*)
(size like *'29inch'*)
[AV-OP grammar]
……
@corp@size@size%and@size
1|2|3|5|bo_or|bo_or|bo_and
@model%and@model@type
1|3|bo_or|4|bo_and
@type@model@pname
1|2|bo_and|3|bo_and
……

[Prefix notation and AV-tree]



To handle negative expressions, we follow the two steps: First, we determine each negation's scope from the input sentence. Second, we insert a "NOT" unary operator into the maximal AV subtree that the negation's scope covers. For example, the input sentence "$_0$ no $_1$ LG $_2$ Electronics $_3$ and $_4$ Philips $_5$" has a negation with [0, 5] scope, and three subtrees: [1, 5] for "LG Electronics and Philips," [1, 3] for "LG Electronics," and [4, 5] for "Philips." Since the negation's scope covers all the subtrees, "NOT" operator is put onto the subtree with [1, 5]. Therefore, (NOT ((corp like '%LG Electronics%') OR (corp like '%Philips%'))) becomes the constraint of the "WHERE" clause.

A query expansion dictionary defines the conceptual sets of values, where the set can definitely determine its values in a given DB, such as "imported product" and "Japanese companies." The conceptual sets are different from user-defined terms in that the meaning of user-defined terms varies with users, such as "large size of TV" and "high-priced audio." For the user-defined terms, we maintain the user's profile for each user, and for the conceptual sets, maintain a query expansion dictionary for the system.

2.3 SQL Query Generation

Domains and DBMSs affect the generation of SQL queries. For multiple domains, our system automatically determines the domain that the user requests; first, find the domains to which each attribute-value node belongs. Second, select one or more domains using the combination of domains from the first step. When two or more domains are chosen, the domains get combined with attributes in the SQL query to be generated, such as "(**saa.**size >= '29inch')" and "(**bb.**corp like '%SONY%')." Where the attribute of question type is ambiguous, the selected domains also help to fix the attribute. We generate an SQL query in the order of question types (SELECT **…** FROM), tables (FROM **…** WHERE), constraints (WHERE **…**), sub-query, and connection with two SQLs.

For supporting session-based dialog, we preserve the SQL query created from the previous questions, and re-generate new SQL query for the current successive question. First, the system checks whether the current question has the same domain as the previous question or not. When the two domains are equal, the previous generated SQL query becomes a constraint part of the current SQL query, for example, "… FROM *(SELECT …)* WHERE … ." Otherwise, a default SQL query[10] will be generated.

[10] "SELECT * FROM *table-name* WHERE 1 (or id>=0)," where id is the primary key.

3 Portability of Languages, Domains, and DBMSs

Natural language interfaces to databases should consider the portability of languages, domains, and DBMSs. Previous systems are short of one or more portability factors; systems with heavy linguistic components have trouble in expanding into other languages, and systems with simple pattern matching are insufficient to deal with multiple domains and DBMSs. On the other hand, our system adopts robust LSP-based language processing and multi-level grammars to structurally analyze the user's query.

Portability of languages: In our system, only both morphological analysis and the resources including dictionaries and grammars are language-dependent. Because the resources include lexico-semantic patterns that represent linguistic characteristics, both the linguistic front-end and the database back-end can be clearly divided. Where the languages to handle are similar to each other, as in word order and linguistic structure, many rules of grammars can be shared without consideration of the specific languages. Like English and Korean[11], however, if the two languages are quite different, then the shared portion naturally decreases. We separate out the language-dependent morphological analysis and the subsequent processes as soon as possible to easily expand to other languages. To add another language, only new morphological analysis and linguistic resources need to be appended to the existing system. Heavy linguistic processes like syntactic and semantic analysis used by the previous systems inevitably delay the point of linguistic separation.

Portability of domains: A new domain with the new DB schema affects both the linguistic front-end and the database back-end. To reduce the influence, our system deals with the domain-related information only in resources and SQL query generation. The other processes, such as morphological analysis, QT/AV processing, and AV-tree construction are all domain-independent. Domain category dictionary and grammars localize the attribute names, but the general category dictionary is independent of domains because it is designed to handle common named entities. In order to manage the multiple domains, SQL query generation should consider domain information. However, a single domain reduces the burden of domain portability because the processing has no relation with the SQL query.

Portability of DBMSs: The format and the descriptive power of SQL queries vary from DBMSs, such as in attribute names, sub-query operation, types of operations, case sensitivity, and constraint syntax. In our system, any linguistic processes and resources do not involve SQL query generation, which eventually increases DBMS portability. SQL query generation has alternatives to produce DBMS-dependent SQL only in some sub-parts, such as sub-query generation and combination with SQL queries. Until an SQL query begins to be generated, current DBMS does not influence any processes and resources.

4 Experiment Results

Pursuing high portability in languages, domains, and DBMSs, we implemented a multi-lingual question answering system on relational databases. The target languages are English and Korean, which are completely different in linguistic structures. Our system dealt with two domains for Korean: first, an audio-video product database with 418 entries automatically populated from an information extraction system [11], and second, a price comparison database with 1964 entries and multiple schemas from a BestBuyer [12] comparison shopping mall. For multiple languages, we manually translated all the Korean entries into English. Oracle 8.0.5 and MySQL 3.23.22 were used as two different DBMSs.

Our system processes the user question from a Web browser, and produces an SQL query. Next, CGI (Common Gateway Interface) sends the query to DBMSs. For the result retrieved from databases, the user can ask a new question or make a context-based refinement of the question.

[11] Korean and Japanese are very similar in that the two are agglutinative languages and have SOV structures, whereas, English and some of the European languages are inflective and have SVO structures.

[12] http://www.bestbuyer.co.kr/mainbbr/index.php3

For training, five graduate students prepare 192 questions for each language (see appendix A). The questions include negation, quantifiers, multiple conjunctions, multiple question types, various lexicography, user-defined terms, synonyms, and many value operators. Table 4.1 shows the current linguistic resources constructed from the training set for both languages.

Resources	English	Korean
Domain category dictionary[13]	2,612 entries	2,847 entries
General category dictionary[14]	63,121 entries	67,280 entries
QT grammar	56 entries	14 entries
AV-TYPE grammar	96 entries	70 entries
AV-OP grammar	94 entries	93 entries

[Table 4.1] Resources for the training set

For the test, we gather 779 unique queries (355 for English and 424 for Korean) and 111 refinement queries (19 for English and 92 for Korean) from the system log for about four months (see appendix A and B). Our system does not fail for the questions because of the LSP-based robustness[15], but some SQL queries with wrong constraints (2.25% for English) are caused by undefined terms, such as "wide TV" and "voice multiplex," and by an illegal unit such as "cm" and "mm." [16] In Korean, the rate of wrong constraints rises to 5.67% that are mainly caused by the irregular transiterations of the foreign words, for example, "텔레비," "텔레비전," "텔레비전," "테레비," "테레비젼," "테레비젼," "티브이," and "티비" for "TV." However, all the above errors can be easily corrected by adding new terms. This phenomenon is also true for multi-level grammars. For a new linguistic expression, we simply decompose it and disperse the components throughout the grammars.

[13] Most of the entries are automatically extracted from target databases.

[14] Reuse the existing dictionaries used for open-domain text question answering

[15] When the system does not find any proper constraint, it produces a default SQL query with null constraint.

[16] Our databases use only "inch" for the size, thus a unit converter needs to cover the errors.

Conclusion

We developed a multilingual question answering system on relational databases and demonstrated high performance and high portability in languages, domains, and DBMSs. LSP-based linguistic processing and multi-level grammars preserve robustness and adaptability without losing the precise interpretation of user queries. In order to overcome previous problems, including the discordance between attribute vocabulary and linguistic processing vocabulary, and the absence of query refinement supporting session-based dialog, we introduced automatic linguistic dictionary construction from database attribute terms, LSP-based linguistic processing, multi-level grammars, and SQL query refinement.

By using lexico-semantic patterns, we separate language-dependent processes from the others at the earliest stage, and use the multi-level grammars to produce sophisticated attribute-value tree structures to connect the attribute vocabulary and the linguistic processing vocabulary. To treat the multiple domains and DBMSs, only SQL query generation and related resources are involved. This minimization of the environment-dependent parts enables our system to be widely ported on multiple environments. Future works include expansion to other languages, including Japanese and Chinese.

References

[1] I. Androutsopoulos, G. Ritchie and P. Thanisch, "MASQUE/SQL – An Efficient and Portable Natural Language Query Interface for Relational Databases," Proc. of the 6th International Conference on Industrial & Engineering Applications of Artificial Intelligence and Expert Systems, 1993.

[2] I. Androutsopoulos, G. Ritchie, and P. Thanisch, "Natural Language Interfaces to Databases – An Introduction," Natural Language Engineering, Vol. 1, No. 1, 1995.

[3] J. Binot, L. Debille, D. Sedlock, and B. Vandecapelle, "Natural Language Interfaces: A New Philosophy," SunExpert Magazine, January, 1991.

[4] P. Cohen, The Role of Natural Language in a Multimodal Interface, Technical Note 514,

Computer Dialogue Laboratory, SRI International, 1991.

[5] R. Dale, H. Moisl, and H. Somers (Eds.), "A Handbook of Natural Language Processing," Marcel Dekker Inc., 2000.

[6] P. Demers, A Lexical Approach to Natural Language Front-end Database, http://www.cs.stu.ca/research/groups/NLL, 1999.

[7] S. Harabagiu, D. Moldovan, M. Pasca, R. Mihalcea, M. Surdeanu, R. Bunescu, R. Girju, V. Rus, and P. Morarescu, "The Role of Lexico-Semantic Feedback in Open-Domain Textual Question-Answering," Proc. of the 39th Annual Meeting and 10th Conference of the European Chapter, 2001.

[8] G. Hendrix, "Natural Language Interface (Panel)," Computational Linguistics, Vol. 8, No. 2, 1982.

[9] H. Jung, G. Lee, W. Choi, K. Min and J. Seo, "A Multi-lingual Question answering System on Relational Databases," Proc. of the 13th Conference on Hangeul and Korean Information Processing (Korean), 2001.

[10] H. Kim, K. Kim, G. Lee, and J. Seo, "MAYA: A Fast Question-answering System Based on a Predictive Answer Indexer," Proc. of the Workshop Open-Domain Question Answering, the 39th Annual Meeting of ACL, 2001.

[11] D. Kim, J. Cha and G. Lee, Learning Information Extraction Patterns for the Web Data Mining, Proc. of the 13th Conference on Hangeul and Korean Information Processing (Korean), 2001.

[12] G. Lee, J. Seo, S. Lee, H. Jung, B. Cho, C. Lee, B. Kwak, J. Cha, D. Kim, J. Ahn, H. Kim and K. Kim, "SiteQ: Engineering High Performance QA System Using Lexico-Semantic Pattern Matching and Shallow NLP," Proc. of the 10th Text REtrieval Conference, 2001.

[13] P. McFetridge, F. Popowich and D. Fass, "An Analysis of Compounds in HPSG (Head-driven Phrase Structure Grammar) for Database Queries," Data & Knowledge Engineering, Vol. 20, 1996.

[14] F. Meng and W. Chu, Database Query Formation from Natural Language using Semantic Modeling and Statistical Keyword Meaning Disambiguation, CSD-TR 990003, University of California, 1999.

[15] A. Mikheev and S. Finch, "Towards a Workbench for Acquisition of Domain Knowledge from Natural Language," Proc. of the 7th Conference of the European Chapter of the Association for Computational Linguistics, 1995.

[16] C. Senturk, Natural Language Interfaces to Databases, In the course of Digital Libraries, E6998-003, 1997.

[17] A. Shankar and W. Yung, gNarLI: A practical Approach to Natural Language Interfaces to Databases, Term Report, Harvard University, 2000.

[18] D. Silberberg and R. Semmel, "Role-Based Semantics for Conceptual-Level Queries," Proc. of the 5th KRDB Workshop, 1998.

[19] M. Templeton and J. Burger, "Problems in Natural Language Interface to DBMS with Examples from EUFID," Proc. of the 1st Conference on Applied Natural Language Processing, 1983.

[20] B. Thompson and F. Thompson, "ASK is Transportable in Half a Dozen Ways," ACM Transactions on Office Information Systems, Vol. 3, No. 2, 1985.

[21] D. Waltz, "An English Language Question Answering System for a Large Relational Database," Communications of the ACM, Vol. 21, No. 7, 1978.

[22] D. Warren and F. Pereira, "An Efficient Easily Adaptable System for Interpreting Natural Language Queries," Computational Linguistics, Vol. 8, 1982.

[23] W. Woods, R. Kaplan, and B. Webber, The Lunar Sciences Natural Language Information System: Final Report, BBN Report 2378, Bolt Beranek and Newman Inc., Cambridge, Massachusetts, 1972.