Conceptual Schema Approach to Natural Language Database Access

In-Su Kang, Seung-Hoon Na, Jong-Hyeok Lee
Div. of Electrical and Computer Engineering
Pohang University of Science and Technology (POSTECH)
Advanced Information Technology Research Center (AlTrc)
San 31, Hyoja-dong, Nam-gu, Pohang, 790-784, R. of KOREA
{dbaisk, nsh, jhlee}@postech.ac.kr
fax: +82-54-279-5699

Abstract

Natural language database interfaces require translation knowledge to convert user questions into formal database queries. Previously, translation knowledge acquisition heavily depends on human specialties such as NLP, DBMS and domain engineering, consequently undermining domain portability. This paper attempts to semi-automatically construct translation knowledge by introducing a physically-derived conceptual database schema, and by simplifying translation knowledge into two structures – class-referring documents and class-constraining selection restrictions. Based on these two structures, this paper proposes a noun translation method that employs an information retrieval framework.

1 Introduction

A natural language database interface (NLDBI) allows users to access database data in a natural language (Androutsopoulos et al., 1995). In a typical NLDBI system, a natural language question is analyzed into an internal representation using linguistic knowledge that normally includes *domain knowledge* to reduce analysis ambiguities. The internal representation is then translated into a target database query by applying *mapping information* (Androutsopoulos et al. 1995) that associates the analysis results with target database structures. This paper uses *translation knowledge* to refer to both domain knowledge and mapping information, because both are required to translate a natural language question into a database query.

Previous approaches can be classified according to the extent that translation knowledge is integrated into linguistic knowledge. Tightly-coupled approaches hard-wire translation knowledge into linguistic knowledge in the form of semantic grammars (Hendrix et al., 1978; Waltz 1978; Templeton and Burger 1983). This approach shows a good performance for a particular domain. However, in adapting to other domains, new semantic grammars should be created with a considerable effort (Allen 1995).

In order to improve domain portability, many researchers have concentrated on isolating translation knowledge from linguistic knowledge through loosely-coupled approaches. These approaches can be further classified according to the extent that question analysis is performed. Syntax-oriented systems (Ballard et al., 1984; Damerau 1985; Lee and Park 2002) analyze questions up to a syntactic level, after which translation knowledge is applied to generate a database query. Logical form systems (Warren and Pereira 1982; Grosz et al., 1987; Alshawi et al., 1992; Androutsopoulos 1993; Klein et al., 1998) interpret a user question into a domain-independent literal meaning level.

Thus, in loosely-coupled approaches, transporting to a new database domain does not need to change linguistic knowledge at all, only tailoring translation knowledge to new domains. Even in this case, however, translation knowledge is difficult to describe. For example, syntax-oriented systems have to devise conversion rules that transform parse trees into database query expressions (Androutsopoulos et al. 1995), and logical form systems should define database relations for logical predicates. In addition, creating these translation knowledge demands considerable human expertise, such as NLP/DBMS/domain specialties. To reduce these manual interventions, many systems employ domain tools (Grosz et al. 1987) to collect domain vocabulary about target database structures, through a sequence of interactive procedures. However, these acquisition processes are passive, and time-consuming. Moreover, such tools cannot be easily adapted to other systems because they are customized to their own systems.

In order to automate translation knowledge acquisition for a new database, this paper attempts to semi-automatically construct translation knowledge by introducing a physically-derived conceptual database schema and by simplifying translation knowledge into two structures – a set of class/value documents having linguistic terms corresponding to domain classes, and a set of selection restrictions on domain classes. Based on these two structures, this paper proposes a noun translation method that employs an information retrieval framework.

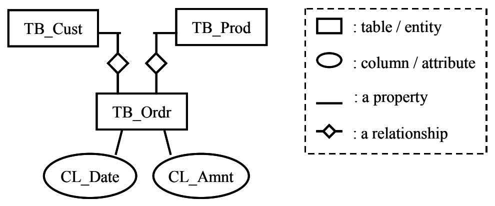
The remainder of this paper is as follows. The next two sections describe terminologies and a conceptual schema used in this paper. Section 4 explains an overview of a conceptual schema approach. Section 5 defines our translation knowledge. Section 6 details a noun translation strategy using translation knowledge, and concluding remarks are given in section 7. For representing Korean expressions, the Yale Romanization is used.

2 Terminologies

In this paper, a database object is defined as either a domain class or a domain class instance. A domain class refers to a table or a column in a database. A domain class instance indicates an individual column value. For example, suppose that a physical database contains two tables, TB_Customer and TB_Employee, and TB_Customer has a column C_cName, and TB_Employee has a column C_eCountry. All these are called domain classes. If the columns C_cName and C_eCountry have 'Abraham Lincoln' and 'France', respectively, as its values, each of these values is called a domain class instance.

A class term is defined as a lexical term referring to a domain class. A value term signifies a term indicating a domain class instance. For instance, the word 'customer' in a user question is a class term corresponding to the above domain class 'TB_Customer'. The word 'Lincoln' is a value term referring to the above domain class instance 'Abraham Lincoln'. In summary, a class term or a value term is used to indicate a word in a user question, and a domain class or a domain class instance is used to refer to a database object, such as a table, a column, or a column value.

3 Physical Entity-Relationship (pER) Schema



pER graph

Domain class	*Linguistic name*	*Definition*
TB_Cust	customer	Persons or companies that order products
TB_Prod	product	NULL
TB_Ordr	order	order information
CL_Date	date of order	dates when customers ordered products
CL_Amnt	amount of order	amount of products that customers ordered

Relationship	*Relationship description*
TB_Cust ⇔ TB_Ordr ⇔ TB_Prod	customers order products customers request orders

pER descriptions

Figure 1. Physical ER (pER) Schema

For a database, a conceptual schema like entity-relationship model structurally resembles a semantic network for the database domain. In addition, its components like entities, attributes, and relationships contain linguistic descriptions, which may bridge between natural language constructions and physical database structures. This paper tries to extract translation knowledge from a conceptual schema. However, a conceptual database schema is not always available. So, we define a physical Entity-Relationship (pER) schema as an approximation of real conceptual schema.

A pER schema is composed of a pER graph and its linguistic descriptions. A pER graph is structurally equivalent to physical database structures, where a node corresponds to a table or a column, and an arc defines a relationship between two tables, or a property between a table and its column. So a node in a pER graph is a domain class. Each node or arc contains linguistic descriptions that are called pER descriptions. As shown in figure 1, there are three kinds of pER descriptions - a linguistic name, definition, and relationship description.

A pER schema is created as follows. First, a logical schema is extracted from a target physical database. This reverse engineering process is automatically performed within a commercially available database modeling tool. The logical schema has the same structure as a physical database. So the logical schema becomes a pER graph. Next, domain experts provide linguistic descriptions for each component of the logical schema according to the following guidelines.

A linguistic name – in a noun phrase
A definition – in a definitional sentence
A relationship description – in a typical sentence including typical domain verbs

The input process is also graphically supported by the database modeling tool, and the pER descriptions can be automatically extracted by the modeling tool.

The term physical ER (pER) schema is used because it is an approximation of the target database's original ER (entity-relationship) schema in the sense that its structures are directly derived from a physical database. A pER graph is later used to generate conceptual query graphs. The pER descriptions have the potential to bridge between linguistic constructions and physical database structures. From pER descriptions, two translation knowledge structures are automatically generated, which will be described in section 5.

4 Conceptual Schema Approach

4.1 Domain Adaptation

Figure 2 shows an NLDBI architecture based on a conceptual schema. There are two processes; domain adaptation and question answering. For a new database domain, domain adaptation semi-automatically constructs translation knowledge. Translation knowledge is divided into two structures: a set of class/value documents corresponding to domain classes, and a set of selection restrictions on domain classes. A class document contains a set of class terms related to a domain class. Class terms are extracted from natural language descriptions of a pER schema.

A value document is created from a set of domain class instances associated with a column. Selection restrictions are also derived from linguistic descriptions of a pER schema.

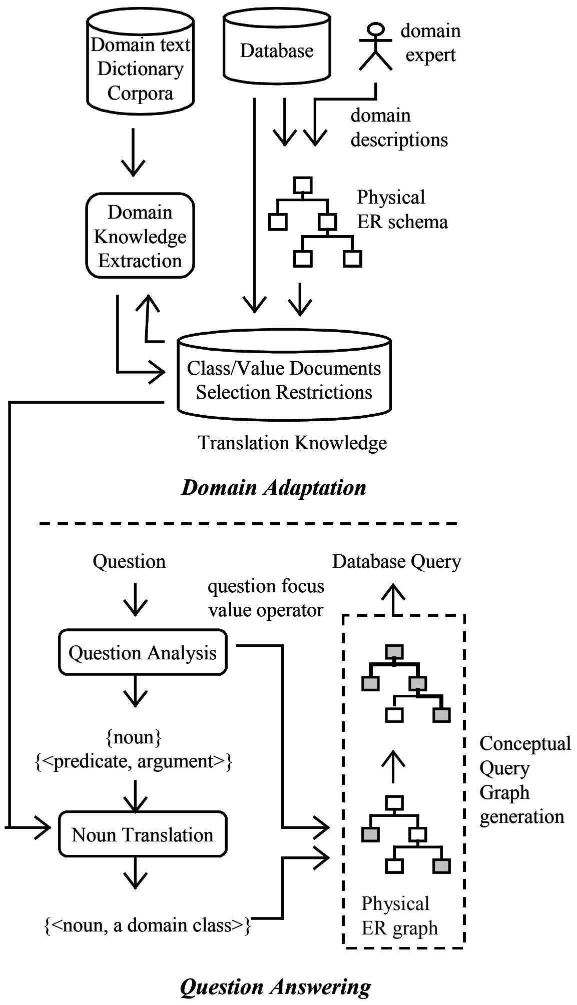


Figure 2. NLDBI architecture

The domain adaptation process may further augment this initial translation knowledge from other resources, such as domain materials, dictionaries, or corpora.

4.2 Question Answering

The question answering proceeds as follows. A user writes his or her information need in a natural language. A user question is then analyzed to produce a set of question nouns and a set of predicate-argument pairs. In Korean, these are obtained after morphological analysis, tagging, chunking, and partial dependency parsing. Question analysis also yields a set of feature-value pairs for each question noun. Among others, essential question features are a question focus and a value operator. These two features determine select-clause items and where-clause operators, respectively, in a final SQL query.

In noun translation, each question noun is considered as an IR query to retrieve lexically or semantically equivalent domain classes in the form of class or value documents. Afterwards, to each question noun holding two or more relevant domain classes, selection restrictions are applied to determine one correct domain class. Noun translation is explained in section 6.

Conceptual query graph generation creates a conceptual query graph (CQG) on the pER graph. First, domain classes produced by noun translation are marked on the pER graph, and question feature-value pairs are attached to the nodes of associated domain classes. On the pER graph, a CQG is searched, which is a connected subgraph connecting all the nodes that are attached with certain feature-value pairs.

The CQG is assumed to represent a domain-dependent question meaning, because nodes of the graph correspond to domain-dependent objects of question nouns, and arcs between nodes represent domain-dependent semantic relations between question nouns. Unlike the logical form method, this approach does not produce an intermediate domain-independent question meaning. Instead, a domain-dependent question meaning is directly represented in the form of a subgraph on a conceptual schema. So a final database query is generated from the graph. Given a CQG, database query generation is trivial. Entity nodes of the CQG go to SQL-from clause, and arcs between entities constitute SQL join operators. An SQL-select clause is obtained from question-focus-attached nodes. All value-operator-attached nodes are combined to create SQL-where conditions.

5 Translation Knowledge

5.1 Translation Knowledge Structures

Any NLDBI system demands translation knowledge, which consists of domain knowledge and mapping information. The former provides an analysis module with ambiguity-reducing devices, such as domain terminologies, domain-dependent selection restrictions, and a domain world model. The latter defines mappings between linguistic analysis results and target database structures. This paper divides translation knowledge into two structures. One is class-referring information, which is a collection of terms that directly refer to each domain class or domain class instance. The other is class-constraining information: a collection of selection restrictions on domain classes. Compared to the previous NLDBI translation knowledge, the first encodes both mapping information and domain terminologies, and the second corresponds to domain-dependent selection restriction. In addition, a pER graph plays the role of a domain world model.

5.2 Class-Referring Translation Knowledge

Formally, class-referring translation knowledge is defined as a set of pairs of *C* and *D*. *C* is a domain class, and *D* is a document that contains terms linguistically referring to *C*. *D* has two types; a class document and a value document. For each domain class, a class document is created from pER descriptions. In addition, for each domain class corresponding to columns, a value document is created from column data for the domain class. These documents are indexed to create a document collection to be used by the later noun translation module.

Class Document

A class document contains a set of lexically synonymous class terms for a domain class. Anticipated class terms are extracted from both linguistic names and definitions in a pER schema. A linguistic name *X* for a domain class is a noun phrase. In Korean, it is a compound noun optionally having a Korean genitive case marker '*uy*' (of). Its general form is $(N^{+}(uy)?\_)^{*}N^{+}$ in a regular expression, where _ is a word boundary and *N* is a simple noun.

A linguistic definition for a domain class is a definitional sentence, so it takes one of the following restricted forms in English translations.

(a kind of) + Y + adjective phrase modifying Y
X|it + be|mean|indicate|... + (a kind of) + Y + adjective phrase modifying Y

Both *X* and *Y* are class term candidates, since a taxonomic relation exists between them. *Y*, which may be also a compound noun, can be easily identified using a few patterns.

Class term extraction proceeds as follows. Given a compound noun, its genitive case markers are deleted, and each of the remaining compound nouns is segmented into a sequence of simple nouns. For example, $N_3N_2uy\_N_1$ is converted into $N_3+N_2+N_1$, where the last noun N_1 is a head of $N_3N_2uy\_N_1$ in Korean, and *uy* is a genitive case marker. Since different combinations of the simple nouns may constitute different question words to refer to the same domain class, a set of head-preserving compound nouns are generated from the simple nouns as follows.

N_3N_2 의$_N_1 \rightarrow N_3N_2+N_1 \rightarrow N_3+N_2+N_1 \rightarrow \{N_3N_2N_1, N_2N_1, N_1\}$

Since a head is an underlying concept of the compound noun, a head noun is preserved for all combinations.

Value Document

For each value term in a user question, an NLDBI system should determine the domain class to which it belongs. This value recognition problem (Templeton and Burger 1983) is critical since, unlike class terms, most value terms are open-ended. In addition, question value terms may take different forms from domain class instances stored in a database. For example, to refer to a database value *sam-seng-cen-ca* (Samsung Electronics) in Korean, users prefer partial forms like *sam-seng* (Samsung) in *sam-seng-ey-se cwu-mwun-han* (… that Samsung ordered).

To support partial matching between question value terms and domain class instances, this paper proposes n-gram value indexing. For each column of a target database, n-gram value indexing generates n-grams of the column data to create a value document. Among column data, linguistic terms are distinguished from alphanumeric terms.

For a linguistic term of k syllables, all-length n-grams from bi-grams to k-grams are generated as index terms of a value document in order to prepare all substrings expected as question value terms. For example, a column value *se-wul-thuk-pyel-si* is processed to generate these n-grams, *se-wul*, *wul-thuk*, *thuk-pyel*, *pyel-si*, *se-wul-thuk*, *wul-thuk-pyel*, *thuk-pyel-si*, *se-wul-thuk-pyel*, *wul-thuk-pyel-si*, *se-wul-thuk-pyel-si*, among which legitimate words as question terms are *se-wul*, *thuk-pyel-si*, *se-wul-thuk-pyel-si*.

On the other hand, generating n-grams for alphanumeric terms causes a severe storage problem. Damerau's method (Damerau 1985) reduces an open-ended set of alphanumeric terms into a closed set of patterns. Thus, it is adopted and slightly modified to include 2-byte characters like Korean. In the modified version, a canonical pattern *P* is defined as follows.

<P> ::= <U>{<U>}
<U> ::= [<C_1>|<C_2>|<N>|<S>][1|2|...|255]
where <C_k> is a sequence of k-byte characters,
<N> is a sequence of numbers,
<S> is a sequence of special characters.

For example, an alphanumeric database value *se-wul*-28@A-*ma* is converted into a canonical pattern, $C_22N2S1C_11C_21$. Next, in order to provide partial matching between patterns, a canonical pattern is decomposed into bi-grams. That is, for $C_22N2S1C_11C_21$, bi-grams $\_C_22$, C_22N2, N2S1, $S1C_11$, C_11C_21, $C_21\_$ are created and stored as index terms in a value document.

Pattern-based n-grams provide considerable storage reduction over storing canonical patterns, since canonical patterns are sliced into smaller n-grams that will have many duplicate n-grams. Hopefully, these n-grams provide partial matching capability even to the arbitrary alphanumeric terms.

5.3 Class-Constraining Translation Knowledge

$$K_v = \{\langle v, C_v \rangle\}, K_{cm} = \{\langle cm, C_{cm} \rangle\}$$

As class-constraining translation knowledge, two types of selection restrictions are defined for domain classes. K_v is a set of selection restrictions between domain verbs and domain classes. K_{cm} is a set of selection restrictions between surface case markers and domain classes. v is a verb appearing in pER descriptions, and C_v is a set of domain classes corresponding to arguments that v governs. cm is a surface case marker appearing in pER descriptions, and C_{cm} is a set of domain classes corresponding to arguments that cm attaches.

K_v and K_{cm} are extracted from predicate-argument pairs that are acquired by parsing pER descriptions. First, each predicate-argument pair is expanded to a triple of <*verb, noun, case marker*>. The *case marker* means a surface case marker of the *noun*. In Korean, the triple <*verb, noun, case marker*> is easily constructed from a predicate-argument pair, since each nominal argument has a surface case marker as a postposition within a word boundary. The second term of a triple is replaced by a *domain class* related to the noun. The modified triple is further divided into <*verb, domain class*> and <*case marker, domain class*>. Next, by merging a set of <*verb, domain class*> having the same *verb*, K_v is produced. Similarly, K_{cm} is obtained by merging a set of <*case marker, domain class*> having the same *case marker*. K_{cm} will be useful for value terms that correspond to different domain classes according to its case marker.

6 Noun Translation

After question analysis, a user question is analyzed into a set of question nouns and a set of predicate-argument pairs. Noun translation utilizes an IR framework to translate each question noun into a probable domain class. First, class retrieval converts each question noun into an IR query and retrieves relevant documents. Here, retrieved documents refer to candidate domain classes for the question noun, because each document is associated with a domain class. Next, class disambiguation selects a likely domain class among the candidate domain classes retrieved by class retrieval using predicate-argument pairs of the user question.

6.1 Class Retrieval

A question noun may be a class term or a value term, and a value term may be a linguistic value term or an alphanumeric value term. To be used as an IR query, these terms are converted into different vector queries, as these terms are differently treated in indexing class or value documents. That is, class terms are converted into word-based terms, linguistic value terms into a list of all-length n-grams, and alphanumeric value terms into a list of pattern-based n-grams. We employ three types of query representations; a conceptual vector for a class term, an all-length n-gram vector for a linguistic value term, and a pattern-based n-gram vector for an alphanumeric value term.

It is straightforward to distinguish whether a question noun is an alphanumeric term. However, it is nontrivial to distinguish between a class term and a linguistic value term, because many domain-dependent class terms are out-of-vocabulary words. So, for a question noun other than an alphanumeric term, class retrieval creates both a conceptual vector and an all-length n-grams vector, and retrieves documents for each query, and merges the retrieved documents. In the following, a conceptual vector representation for class terms is described.

If we simply convert a class term into a single term vector, it may cause a severe word mismatch problem (Furnas el al., 1987). Thus, the question noun is generalized to concept codes, which are then included in a vector query. Unfortunately, this method may risk obtaining mistaken similarity values if the correct concepts of the two terms are not similar while incorrect concepts of the two terms are similar. However, considering that domain terminologies show marginal sense ambiguities (Copeck et al., 1997), this concern will not be critical.

A query-document similarity is computed as follows.

$$Similarity(Q, D) = argmax_t\, W_Q(t) * W_D(t)$$

It simply selects the maximum value among weights of each matching term t. The reason is that, because all query terms belong to one homogeneous term group that originates from one lexical query term, the similarity between a query and a document means the best of similarities between the homogeneous group of a query term and homogeneous groups of several document terms.

6.2 Class Disambiguation

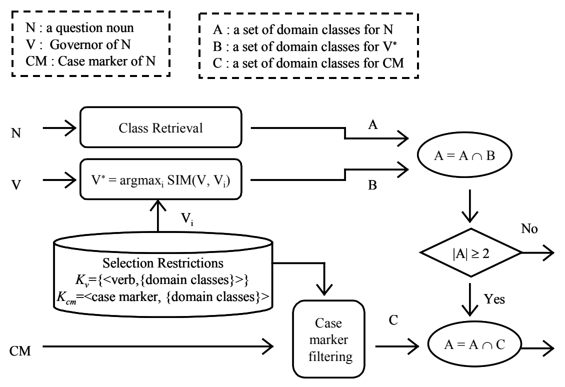

N : a question noun
V : Governor of N
CM : Case marker of N
A : a set of domain classes for N
B : a set of domain classes for V*
C : a set of domain classes for CM
N
V
CM
Class Retrieval
V* = argmax_i SIM(V, V_i)
V_i
Selection Restrictions
K_v={<verb,{domain classes}>}
K_cm=<case marker, {domain classes}>
Case marker filtering
A
B
C
A = A ∩ B
|A| ≥ 2
No
Yes
A = A ∩ C

Figure 3. Class Disambiguation

When question nouns are translated into domain classes, two types of ambiguities occur. Class term ambiguity occurs when a class term in a question refers to two or more domain classes. This ambiguity mostly results from general attributes that several domain entities share. For example, a question noun ‘address’ can refer to any ‘address’ attribute that the two entities ‘customer’ and ‘employee’ have at the same time. A value term ambiguity occurs when more than two domain classes share at least one domain class instance. Hence, date or numeric expressions almost always cause value term ambiguity. In particular, in an air flight domain, country names or city names will be shared by many domain classes, such as the location of departure and the location of arrival.

Class retrieval reduces the translation equivalents of each question noun to lexically or semantically equivalent domain classes. However, the above two ambiguities still remain after class retrieval. Class disambiguation resolves these ambiguities using class-constraining translation knowledge of K_v and K_{cm}. Disambiguation procedures proceed in two stages, as shown in figure 3.

In the first stage, for each question noun with two or more domain classes after class retrieval, K_v is searched to find a domain verb that is the most similar to the head verb of the question noun. The SIM value between two lexical words is the maximum of concept similarity values between all possible concept pairs of the two lexical words. Let B represent the set of domain classes associated with the domain verb, and let A be the set of domain classes retrieved by class retrieval for the question noun. Then, A is replaced by A intersection B. The effect is to reduce ambiguities by removing from A inconsistent domain classes that is not expected by a governor of the question noun.

The second stage takes the remaining ambiguities after applying K_v. K_{cm} is searched to find the same surface case marker as that of the question noun, and let C be the set of domain classes associated with the case marker. Then, A is further replaced by A intersection C. The effect is to select from A only the domain classes to which the case marker can attach.

For example, consider the following question.

Q1: sam-seng-ey-se *cwu-mwun-han cey-phwum-un ?*
E1: Show me products (cey-phwum) that Samsung(sam-seng) ordered (cwu-mwun-han) ?

A word *sam-seng-ey-se* consists of a root *sam-seng* and a postpositional case marker *ey-se*. Suppose that the question noun *sam-seng* retrieves three ambiguous domain classes {TB_Supplier, TB_Customer, TB_Shipper} by class retrieval. Then, using a governor *cwu-mwun-ha* of *sam-seng*, K_v is searched to find <*cwu-mwun-ha*, {TB_Customer, TB_Product, TB_Order.Amount, TB_Order.Date} >, and reduce ambiguity as follows.

A = {TB_Supplier, TB_Customer, TB_Shipper}
B = {TB_Customer, TB_Product, TB_Order.Amount, TB_Order.Date}
A = A ∩ B = {TB_Customer}

In this case, K_{cm} is not used. As another example, consider this question.

Q2: se-wul-ey-se len-ten-kka-ci pi-hayng-si-kan-un ?
E2: Show me the flight duration (pi-hayng-si-kan) from(ey-se) Seoul(se-wul) to(kka-ci) London(len-ten) ?

By class retrieval, a question noun *se-wul* will retrieve two domain classes {TB_Flight.Departure, TB_Flight.Arrival}. Unlike Q1, B will be empty since Q2 does not provide any verb. Then, using a case marker *ey-se*, K_{cm} is searched to find < *ey-se*, {TB_City, TB_Country, TB_Flight.Departure}>, and reduce ambiguity as follows.

A = {TB_Flight.Departure, TB_Flight.Arrival}
C = {TB_City, TB_Country, TB_Flight.Departure }
A = A ∩ C = {TB_Flight.Departure}

7 Conclusion

To effectively deal with the domain portability problem, this paper proposed the conceptual schema approach, which depends on the following three main components that differ from previous approaches.

The first is an introduction of a physical ER schema, which is easily created from a target database itself by domain experts with the help of a database modeling tool. The schema is used for capturing domain-dependent question meaning, because semantic constraints among domain objects are represented in the graph part of the ER schema. The second is the automatic construction of translation knowledge from a physical ER schema. To accomplish this, we defined two types of translation knowledge structures: a set of class-referring documents and a set of class constraining selection restrictions. The construction process requires only a shallow analysis of linguistic descriptions for a physical ER schema. The third is a noun translation strategy based on an information retrieval framework, where question nouns are associated with domain classes, lexically or semantically.

In future, we will extend the current translation knowledge from other resources, such as domain materials, dictionaries, and corpora.

Acknowledgements

This work was supported by the Korea Science and Engineering Foundation (KOSEF) through the Advanced Information Technology Research Center(AITrc).

References

Allen J. 1995. *Natural Language Understanding*. Redwood City, CA:Benjamin Cummings.

Alshawi, H., Carter, D., Crouch, R., Pulman, S., Rayner, M., and Smith, A. 1992. CLARE – A Contextual Reasoning and Cooperative Response Framework for the Core Language Engine. *Final report*, SRI International.

Androutsopoulos, I. 1993. Interfacing a Natural Language Front-End to Relational Database. *Master's thesis,* Technical Report 11, Department of Artificial Intelligence, University of Edinburgh.

Androutsopoulos, I., Ritchie, G.D., and Thanisch, P. 1995. Natural Language Interfaces to Databases – An Introduction. *Natural Language Engineering,* 1(1):29-81.

Ballard, B.W., Lusth, J.C., and Tinkham, N.L. 1984. LDC-1: A Transportable, Knowledge-Based Natural Language Processor for Office Environments. *ACM Transactions on Office Information Systems* 2(1):1-25.

Copeck, T., Barker, K., Delisle, S., Szpakowicz, S., and Delannoy, J.F. 1997. What is Technical Text?. *Language Sciences* 19(4):391-424.

Damerau, F. 1985. Problems and Some Solutions in Customization of Natural Language Database Front Ends. *ACM Transactions on Office Information Systems* 3(2):165-184.

Furnas, G.W., Landauer, T.K., Gomez, L.M., and Dumais, S.T. 1987. The vocabulary problem in human-system communication. *Communications of the ACM* 30(11):964-971.

Grosz, B.J., Appelt, D.E., Martin, P.A., and Pereira, F.C.N. 1987. TEAM: An Experiment in the Design of Transportable Natural-Language Interfaces. *Artificial Intelligence* 32(2):173-243.

Hendrix, G.G., Sacerdoti, D., Sagalowicz, D., and Slocum, J. 1978. Developing a Natural Language Interface to Complex Data. *ACM Transactions on Database Systems* 3(2):105-147.

Klein, A., Matiasek, J., and Trost, H. 1998.. The treatment of noun phrase queries in a natural language database access system. *Proceedings of the COLING-ACL'98 workshop on the computational treatment of nominals*, Montreal, Quebec, pp.39-45.

Lee, H.D., and Park, J.C. 2002. Interpretation of Natural language Queries for Relational Database Access with Combinatory Categorial Grammar. *International Journal of Computer Processing of Oriental Languages* 15(3):281-304.

Templeton, M., and Burger, J. 1983. Problems in Natural Language Interface to DBMS with Examples with EUFID. *Proceeding of the 1st Conference on Applied Natural Language Processing,* Santa Monica, California, pp.3-16.

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

Warren, D., and Pereira, F. 1982. An Efficient Easily Adaptable System for Interpreting Natural Language Queries. *Computational Linguistics* 8(3-4):110-122.