Simple+Robust = Pragmatic : A Natural Language Query Processing Model for Card-type Databases

Seigou Arita Hideo Shimazu Yosuke Takashima

C&C Information Technology Research Laboratories
NEC Corporation
1-1,Miyazaki 4-chome, Miyamae-ku,Kawasaki,Kanagawa 216 Japan
arita%joke.cl.nec.co.jp@uunet.uu.net

Abstract

Real users' queries to databases written in their natural language tend to be extra-grammatical, erroneous and, sometimes just a sequence of keywords. Since most conventional natural language interfaces are *seminatural*, they cannot treat such real queries very well. This paper proposes a new *natural* language query interpretation model, named SIMPLA. Because the model has a keyword-based parsing mechanism, it is very robust to cope with extra-grammatical sentences. The strong keyword-based parsing capability is very dependent upon its target database's being a "card"-type. SIMPLA provides several operators to define peripheral knowledge, regarding the target database. Such peripheral knowledge is stored *virtually* in parts of the target "card"-type database. Since the target database with the peripheral knowledge remains "card"-type, SIMPLA does not decrease its robust natural language processing capability, while it embodies the ability to respond to questions concerning peripheral questions.

1. Introduction

As the number of commercial databases increases, the expectations of ordinary people, with regard to *pragmatic* natural language (NL) interface to databases, are increasing. Though many research efforts on user interfaces, including menu systems and friendly command language, are running, they are still *unnatural* to ordinary people.

Many research efforts on NL interfaces have also been implemented for more than 20 years (Winograd, 1977)(Simmons, 1970)(Hendrix et al., 1978)(Tennant, 1981). Some of them, like INTELLECT (Shneiderman, 1987), have been used in real business applications. However, they still have only *seminatural* language processing capabilities. Most conventional NL interface systems cannot treat queries that are written really *naturally*. Real users' queries to databases, written in their natural language, tend to be extra-grammatical, erroneous and, sometimes just a sequence of keywords. A few research efforts (Carbonell et al., 1984) have been struggling with extra-grammaticality. However, they have not yet succeeded in coping with such real queries.

This paper proposes a new *natural* language query interpretation model. Because the model has a keyword-based parsing mechanism, it is very robust to cope with extra-grammatical sentences. The proposed natural language interface model, **SIMPLA**(SIMPle Language Analyzer), has the following characteristics:

- Parsing is keyword-based:
  SIMPLA extracts only keywords from an input sentence, and generates its interpretation from the extracted keywords. Therefore, the parsing is very robust to extra-grammatical expressions. Even a sequence of keywords, as an input sentence, can be correctly interpreted.

- Operators are provided to define peripheral knowledge and to put it into a target "card"-type database virtually:
  SIMPLA can retrieve appropriate data from its target "card"-type database, as the response to a natural language query. NL interface must respond to queries which are not just a direct data retrieval of the target database, but are concerning questions. To interpret and reply

to such extended questions, the NL interface must hold peripheral knowledge for the target database. In SIMPLA, several operators are provided to define such knowledge. However, the strong capability of SIMPLA's keyword-based parsing is very dependent upon its target database's being "card"-type. So, SIMPLA holds such knowledge as if it were placed in a part of the target "card"-type database. It looks like the target databases are extended, but the extension is *virtual*. The authors call the extended database a **virtual database**. Since the form of the target database remains "card"-type, SIMPLA does not decrease its robust natural language processing capability, while SIMPLA embodies the ability to respond to peripheral questions.

Section 2 shows an example that provides the concept regarding how SIMPLA processes a query. Section 3 describes notions of label base and vocabulary space. Section 4 illustrates the algorithm for interpreting queries. Section 5 explains knowledge representation for domain-oriented thesaurus through the virtual extension of target databases.

2. Basic SIMPLA Idea

SIMPLA's interpretation mechanism is novel. SIMPLA analyzes queries using much simpler grammar than that involved in conventional linguistic approaches.

First, look at the process for interpreting the following sample query towards a sample database "world handbook", shown in Figure 1. It provides the concept regarding how SIMPLA interprets queries (Arita et al, 1991).

Nation	Area	Population	Capital	Language
Korea	99016	42380	Seoul	Korean
Canada	9976139	26250	Ottawa	English,French
Japan	377801	123120	Tokyo	Japanese

Figure 1: World handbook

S1: "Where is the capital of Canada?"

When S1 is given to SIMPLA, it extracts only keywords from the sentence. Here, the keywords are "capital" and "Canada". "Capital" is an attribute name. "Canada" is a value of the "nation" attribute in a record. SIMPLA constructs the meaning of this sentence, using only these bits of information. SIMPLA searches the target database for the records whose "nation" attribute has the value, "Canada", then, gets the value of the "capital" attribute for the extracted records. In this case, a record, which includes the attribute value "Canada", is the second record. A value for the attribute name "capital" for the second record is "Ottawa". So, "Ottawa" is an output.

Like the above example, most of those queries fall into the SELECT-FROM-WHERE type queries in SQL. SIMPLA regards all the queries as SELECT-FROM-WHERE instructions.

Thus, the process has been implemented in a keyword based manner. The above process is tolerant to extra-grammatical queries. The algorithm is far more easily implemented than, the ones using conventional linguistic approaches.

SIMPLA's actual mechanism is more complex. It includes the notion of a virtual database. The virtual database is a virtually extended target database with thesaurus. First, SIMPLA analyzes an input query as a query targeted to the virtual database, and generates its internal representation. Then, it translates the representation to the real query command to the actual database management system.

3. Label Base And Vocabulary Space

SIMPLA regards a query as a sequence of *descriptions on a relationship between an attribute name and its value*. For instance, the S1 sentence,"Where is the capital of Canada" are regarded as a sequence of descriptions: (1) "capital" is an attribute name with no attribute value assigned in the query. (2) "Canada" is equal to the value of "nation". On this semantic, which pairs of an attribute name and its value are existing in a target database form the most basic information. In SIMPLA, such information is prepared in **label base** and **vocabulary space**.

Label base is a set of **basic pairs** of binary relation:

```
basic_pair(Attribute, Value).
```

Here,"Attribute" and "Value" indicate an attribute name and an attribute value in a target database, respectively. Vocabulary space is a set of **vocabulary pairs** of binary relation :

```
vocabulary_pair(VirtualAttribute,
VirtualValue).
```

Here, "VirtualAttribute" and "VirtualValue" indicate an attribute name and an attribute value in a virtual database, respectively. Correspondences between basic pairs in label base and vocabulary pairs in vocabulary space are given in **anchor**. For example, label base, vocabulary space and anchor of the "world handbook" in Figure 1 are shown in Figure 2 and in Figure 4, respectively.

Virtual database is an image of the database, which is seen by SIMPLA, with vocabulary space in place of label base (Figure 5). SIMPLA's parser refers to vocabulary space in order to determine that an input word is either an attribute name or an attribute value. An attribute name and an attribute value respectively appear as a left component and a right component in a vocabulary pair in vocabulary space. This is the formal definition for an attribute name and an attribute value in SIMPLA:

```
attribute_name(AttributeName)
:- vocabulary_pair(AttributeName, _).

attribute_value(AttributeValue)
:- vocabulary_pair(_,AttributeValue).
```

For example, with vocabulary space in Figure 4 in place of label base in Figure 2, the target database seems to have "east/west" and "population density" attributes.
Because, the `attribute_name('east/west')` and `attribute_name('population density')` hold in that definition. So, the database appears to have more attributes than really existing ones ,as "virtual world handbook", shown in Figure 3.

4. SIMPLA's Structure

SIMPLA's structure is shown in Figure 6. First, Parser translates the user's query into **virtual form**. Second, Realizer translates the virtual form to **real form**. Finally, Generator generates a raw retrieval form from the given real form , which is executable by the target database system.
Both virtual form and real form are sequences of **units**. Unit shows the relation between an attribute name and value.

Parser

Parser translates a query to a virtual form, with reference to vocabulary space, as follows: (1). To extract a sequence of relational words, attribute names and attribute values from the query through reference to vocabulary space. (2). To apply simple

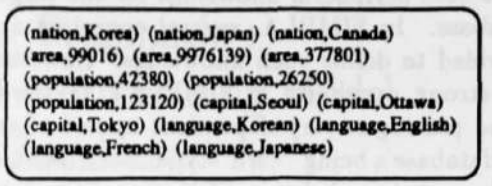

Figure 2: Label base

Nation	East/west	Area	Population	Pop.density	Capital	Language
Korea	East	99016	42380	0.428	Seoul	Korean
Canada	West	9976139	26250	0.003	Ottawa	English,French
Japan	East	377801	123120	0.326	Tokyo	Japanese

Figure 3: Virtual world handbook

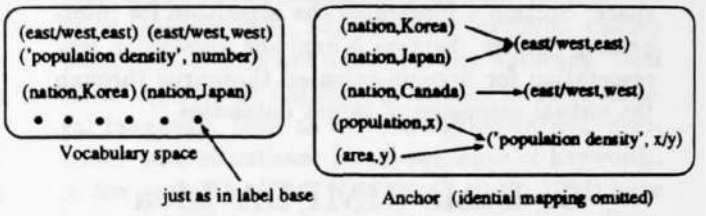

Figure 4: Vocabulary space and anchor

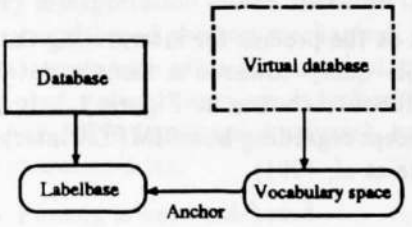

Figure 5: Virtualization by vocabulary space and anchor

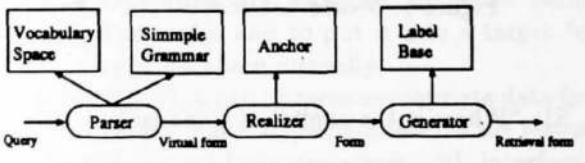

Figure 6: SIMPLA's structure

```
parse([]) --> [].
parse([Unit | Units]) --> gen_unit(Unit), parse(Units).

gen_unit(unit(Type,Name,Value)) -->
   [attribute_name(Name),attribute_value(Value),relational(Type,R)],
   {vocabulary_pair(Name,Value)}.      (1)
gen_unit(unit(Type,Name,Value)) -->
   [attribute_value(Value),relational(Type,R),attribute_name(Name)],
   {vocabulary_pair(Name,Value)}.      (2)
gen_unit(unit(eq,Name,Value))      -->
   [attribute_name(Name),attribute_value(Value)],
   {vocabulary_pair(Name,Value)}.      (3)
gen_unit(unit(eq,Name,Value))      -->
   [attribute_value(Value),attribute_name(Name)],
   {vocabulary_pair(Name,Value)}.      (4)
gen_unit(unit(eq,Name,Value))      -->
   [attribute_value(Value)],
   {vocabulary_pair(Name,Value)}.      (5)
gen_unit(unit(eq,Name,_))      -->
   [attribute_name(Name)],
   {vocabulary_pair(Name,_)}.          (6)
```

Figure 7: SIMPLA's simple grammar

grammar to the sequence in order to obtain the virtual form. **Relational word** describes a relation between an attribute name and its vale.

```
relational(gtr, ''greater than'').
relational(sml, ''smaller than'').
...
```

The simple grammar is a set of rules used to transform a sequence of relational words, attribute names and attribute values into a virtual form. Some of these rules are shown in Figure 7, in DCG style.

For instance, gen_unit (1) means: if attribute name **Name**, attribute value **Value** and relational word **R** for category **Type** appear at the top of the sequence and if **(Name, Value)** appears in the vocabulary space as a vocabulary pair, then generate **unit(Type, Name, Value)**. The rest of the sequence are processed in the same way. The number of applied rules are equal to the number of generated units. Parser's output is virtual form, which is a sequence of those units.

For instance, look at the process of parsing the following query:

S2:"List languages in the East."

First, the query is filtered with vocabulary space into a sequence:

```
[attribute_name(language),
attribute_value(east)].
```

Second, the simple grammar is applied to the sequence. The gen_unit (6) matches the top of the sequence:

```
gen_unit(virtual_unit(eq, language, _))
← [attribute_name(language)]
```

and vocabulary pair (language, _) belongs to the vocabulary space [1]. So, unit **unit(eq, language, _)** is generated. Similarly, by gen_unit (5) is applied to the rest of the sequence **[attribute_value(east)]**, unit **unit(eq, east/west, east)** is generated. Thus, Parser outputs virtual form:

```
[unit(eq, language, _), unit(eq,
east/west, east)].
```

Realizer

Since a virtual form may include "virtual" attribute names or values (as "east/west" in the above example), virtual form must be realized to real form by Realizer. Realization of virtual form to real form is accomplished by realizing each unit in virtual form. The unit realization is driven by anchor, which is a map between basic pairs in label base and vocabulary pairs in vocabulary space. For instance, the virtual form [unit(eq, language, _), unit(eq, east/west, east)] is realized to a real form [unit(eq, language, _), unit(eq, nation, korea), unit(eq, nation, japan)], because anchor in Figure 4 holds a mapping

[1] "_" indicates an uninstantiated variable, just as in Prolog

Anchor: (nation, Korea), (nation, Japan)
→ (east/west, East)

Generator

Finally, Generator translates real form into a retrieval form, which is executable by the target database system. Unit in real form generates Select-clause, if its attribute value remains an uninstantiated variable. Other units in real form generate Where-clause. For instance, real form [unit(eq, language, _), unit(eq, nation, korea), unit(eq, nation, japan)] is translated into the retrieval form "SELECT language FROM world_handbook WHERE nation = "Korea" OR nation = "Japan" " , if represented in SQL.

SIMPLA doesn't completely interpret all queries correctly. However, with emphasis on practical use, SIMPLA's target is to interpret most practical queries immediately and robustly, avoiding make the model too naive and too large regarding the cost for processing very complicated and unusual queries.

5. Thesaurus

This section describes how SIMPLA virtualizes a database. Virtualizing a database is just generating vocabulary space and anchor associated to the database. SIMPLA provides several operators to define a schema for a virtual database, as the extension of an original database. These operators are:

Name index operator
`name_index(Attribute, AttributeName)`. To name an attribute `Attribute` in a database as `AttributeName`.

Value index operator
`value_index(AttributeName, Value, ValueName)`. To name a value `Value` for an attribute `AttributeName` as a `ValueName`.

Grouping operator
`grouping(A, {V1,...,Vn}, NewA, NewV)`. To register a set of some attribute values {`V1`, ..., `Vn`} for an attribute name `A` as a new attribute value `NewV` for a (new) attribute name `NewA`.

Compound operator
`compound(A, {A1, ..., An}, ψ)`. To define a new attribute `A` using already defined attributes `A1`, ..., `An` in a database, where ψ is an expression that defines `A`.

Using a name index operator and a value index operator, the natural language interface designer can assign natural language fragments to each corresponding bit of data in the database. By using a grouping operator, the designer can align the database attributes into a hierarchy. By a compound operator, the designer can form associations for the attributes. For example, regarding the "World handbook" database(Figure 1), the operators in Figure 8 can be implemented, so that the virtual database "Virtual world handbook"(Figure 3) is obtained.

```
name_index(Any, Any).                                          (10)
value_index(AttributeName, AnyValue, AnyValue).                (11)
group(nation, {korea, japan}, 'east/west', east).              (12)
group(nation, {canada}, 'east/west', west).                    (13)
compound('population density', {population, density}, /).      (14)
```

Figure 8: The "virtual world handbook" definition

Virtualizing Database by Vocabulary Space

Here, virtualizing operators, defined in the previous section, are implemented as operations on vocabulary space.

First, label base is constructed using name index operator and value index operator. In the "World handbook" case, operators (10) and (11) have been applied, in Fig8. Because they specify nothing special, label base consists of all pairs of attribute names and attribute values. That is, the label base for the "World handbook" is as shown in Figure 2. An initial state of vocabulary space is equal to label base, when the anchor is trivial.

Anchor: vocabulary_pair(A, V)
→ basic_pair(A, V)

The grouping operator and the compound operator modify vocabulary space and anchor.

The grouping operator **grouping(A, {V1,...,Vn}, NewA, NewV)** operates on vocabulary space and anchor, as follows: (1). To add vocabulary pair (**NewA**, **NewV**) to vocabulary space VS. (2). To define the image for anchor *Anchor* versus the above vocabulary pair , as follows:

Anchor: vocabulary_pair(NewA, NewV)
→ basic_pair(A, $\{V_1, \ldots, V_n\}$)

(The right-hand side means that attribute name A takes any one of $V_1, \ldots, V_n$ as a value).

In "World handbook" case, operator (12) `grouping(nation, {korea, japan}, east/west, east)` is applied in Figure 8. It adds vocabulary pair (east/west, east) to the vocabulary space VS. The anchor image for this pair is as follows:

Anchor: vocabulary_pair(east/west, east)
→ basic_pair(nation, {korea, japan})

The compound operator `compound(A, {A1, ..., An}, ψ)` operates on vocabulary space and anchor as follows: (1). To add vocabulary pair (`A`, `X`) having the first argument `A` as an attribute name and a variable `X` as an attribute value to vocabulary space VS. (2). To define the image for anchor *Anchor* versus the above vocabulary pair , by the rest of the arguments as follows:

Anchor: vocabulary_pair(A, X)
→ basic_pair(ψ(A1, ..., An), X)

In "World handbook" case, operator (14) `compound('population density', {population, density}, /)` is applied in Figure 8. It adds vocabulary pair ("population density", X) to vocabulary space VS. The anchor image for this pair is as follows:

Anchor: vocabulary_pair('population density', X)
→ basic_pair(population/density, X)

After processing all operators displayed in Figure 8, SIMPLA gets vocabulary space and anchor, as illustrated in Figure 4.

Conclusion

In this paper, the authors described SIMPLA, a new natural language query processing model, and its implementation. SIMPLA has a very simple parsing mechanism augmented with the notion of a virtual database. SIMPLA can interpret extragrammatical sentences as well as ordinary natural language sentences. It even accepts just a sequence of keywords, as an input sentence. Therefore, SIMPLA can be placed among conventional natural language processing model, command language interpreter, and keyword-based information retrieval model.

Of course, the linguistic capability for SIMPLA is not as strong as conventional NL systems, which accept predicted *seminatural* language descriptions. However, in the real applications, SIMPLA's hybrid interpretation ability is more *pragmatic* for conventional natural language processing model, command language interpreter, and keyword-based information retrieval model. Users can generate queries in multi-style. If a user can not generate a natural language query, which SIMPLA can interpret correctly, he/she just makes a sequence of keywords, instead. SIMPLA may accept the sequence appropriately.

The SIMPLA's response time is faster than that for a conventional NL interface, because SIMPLA's processing work load is far lighter, compared with the fully parsing approach. According to early experimental results, the response time for a query to a database, whose size is more than 2500 records, was less than two seconds, implemented on Quintus-Prolog, on Sparc-station-1.

Giving up an attempt to process very complicated and unusual queries, SIMPLA has gained robust ability to interpret most of simple queries immediately and correctly. Now, to interpret even those unusual queries efficiently, the authors are trying to combine Case-based method with this approach (Shimazu et al, 1991)(Shimazu et al, 1992).

References

ARITA, S.; SHIMAZU, H.; and TAKASHIMA, Y. 1991. Simple Natural Language Interface Model, In Proc. of the 5th Annual Conference of JSAI.

CARBONELL, J.G.; and HAYES, P.J. 1984. Recovery strategies for parsing extragrammtical language, Technical Report CMU-CS-84-107, Dept. of Computer Science, CMU.

HENDRIX, G.G.; SACERDOTI, E.D.; SAGALOWICZ, D., and SLOCUM, J., 1978. Developing a Natural Language Interface to Complex Data, In ACM Trans. on Database Systems.

SHIMAZU,H.; ARITA,S.; and TAKASHIMA,Y. 1992. Design Tool Combining Keyword Analyzer and Case-based Parser for Developing Natural Language Database Interfaces, Proc. of COLING-92.

SHIMAZU,H.; and TAKASHIMA,Y. 1991. Acquiring Knowledge for Natural Language Interpretation Based On Corpus Analysis, Proc. of IJCAI-91 Natural Language Learning Workshop.

SHNEIDERMAN, B. 1987. Designing the User Interface, Addison-Wesley Pub.

SIMMONS, R.F. 1970. Natural Language Question-Answering Systems, CACM, Vol. 13, Jan.

TENNANT, H. 1981. Natural Language Processing, Petrocelli Books.

WINOGRAD, T. 1977. Understanding natural Language, Academic Press.