自然言語処理 89－7
（1992. 5. 15）

簡易型自然言語インタフェースモデル
― データベースの仮想化に基づくアプローチ

有田正剛　島津秀雄　高島洋典
日本電気（株）C&C 情報研究所

実用的な自然言語インタフェースにとって、何よりも重要なのは、強健なパーザと充実したシソーラスである。ところが、上の2点の要件は、単独の問題としても、実現のむずかしいものであるが、さらに、それらはトレードオフの関係にある。本稿の提案する自然言語インタフェースモデル SIMPLA(SIMPle Language Analyzer) は、これら2つの要件を、その間のトレードオフを解消しつつ、満足することのできる新しいタイプの自然言語インタフェースを実現する。SIMPLA は、キーワードベースのパージングをおこない、また、あたかも対象のデータベースの一部に新しいデータを付け加えるかのようにしてシソーラスを持つことによって、パーザに負担をかけることなく、シソーラスを実装できる。

A Simple Natural Language Interface Model
— An Approach with A Virtualized Database

Seigo Arita　Hideo Shimazu　Yosuke Takashima
C&C Information Technology Research Laboratories
NEC Corporation

Improtant problems in practical natural language interface are to construct a robust parser and to implement sufficient thesaurus. They are difficult not only respectively but also they are in trade-off relation. This paper proposes a new natural language interface model SIMPLA. SIMPLA realizes the natural language interface that overcomes the two problems with resolving the trade-off between them. Because the model has a keyword-based parsing mechanism, it is very robust to cope with extra-grammatical sentences. Peripheral knowledge is stored *virtually* in the part of the target database, so SIMPLA does not decrease its robust natural language processing capability, even with complicated knowledge as thesaurus.

1 はじめに

現在、多くのオンラインデータベースが、パソコン通信を通して、日常的に利用されつつある。今や日本のオンラインデータベースは3000を越え、PC-VANの会員は35万人を数える。ワープロの普及にともないキーボードアレルギーも過去のものとなりつつある。今後は特別な部署やサーチャーが代行検索するのではなく、我々ビジネスマンが欲しい情報をデータベースから直接検索し入手する時代になると思われる。しかしながら、現在のデータベースシステムのインタフェースは一般利用者にとって使いやすいとは言えない[10][9]。特別な訓練を必要としないインタフェースとして、自然言語インタフェースへの潜在的需要は大きい[3]。

実用的な自然言語インタフェースにとって、何よりも重要なのは、強健なパーザと充実したシソーラスである。データベースへの問い合わせ文には省略されている単語が多く、通常の文章には出現しない記号が現れたりする。そもそもきちんとした文の体裁をとるのは珍しいとさえいえる。したがって、自然言語インタフェースのパーザは非文法性に対して強健であることが必要とされる。また、利用者の問い合わせは、対象のデータベースに直接には存在しない属性名や属性値を連想していることが多い。例えば、利用者は「住所」の属性に「県名」が書いてあるとすると「地方名」で聞きたくなる。実用的な自然言語インタフェースには充実したシソーラスが必要である。

上の2点の要件は、単独の問題としても、実現のむずかしいものであるが、さらに、問題を複雑にしているのは、パーザの強健さとシソーラスの充実とはトレードオフの関係にあることである[5]。シソーラスを充実させれば、その分パーザの処理に負担がかかってしまう。

本稿の提案する自然言語インタフェースモデルSIMPLA(SIMPle Language Analyzer)は、これら2つの要件を、その間のトレードオフを解消しつつ、満足することのできる新しいタイプの自然言語インタフェースを実現している[1][2]。

SIMPLAでは、利用者の発するデータベース検索命令は、論理的な構造についてはパターン化しているとの分析に基づき、キーワードベースのパージングをおこなう。よって、パーザは非文法性に対して強健である。また、SIMPLAは、あたかも対象のデータベースの一部に新しいデータを付け加えるかのようにしてシソーラスを持つ。これによって、シソーラスと対象データベース中のデータはまったく同様に扱えるので、パーザに負担をかけることなく、シソーラスを実装できる。また、シソーラスの獲得は対象データベースを仮想的に拡大するという文脈において行なわれるので、自然で容易である。これは、システムの移行性に寄与する[4][6]。

2節では、SIMPLAの基本的な考え方を例によって示す。3節では、SIMPLAの用いる問い合わせ文の意味表現について述べ、4節では、それらを用いてSIMPLAの解釈アルゴリズムを述べる。5節では、対象データベースの仮想化の文脈で、シソーラスが獲得されることを示す。

2 SIMPLAの基本的な考え方

SIMPLAの問い合わせ文の解釈アルゴリズムは特徴的である。通常の言語学的な文法を用いるのではなく、遥かに簡易な文法を用いて問い合わせ文を解析する。これは、利用者の命令の論理的な構造はパターン化していることを踏まえ、問い合わせ文を(対象データベースのスキーマに在る)**属性とその値との関係の宣言の連なり**であると見なすことで可能となる。

最初に、SIMPLAの問い合わせ文を解釈する基本アルゴリズムの働きを示すために、図1のサンプルデータベース「政治家データベース」に対する以下の問い合わせ文の処理を見る。[2]

名前	政党	役職	就任	退任
海部	自民党	党首	1989	1991
大石	公明党	党首	1988	1989
土井	社会党	党首	1987	1989

図1: 政治家データベース

「社会党の党首の名前は？」

このアルゴリズムが他の自然言語処理モデルあるいはシステムと異なる重要な点は、**対象データベースのスキーマにある属性の名前や値をキーワードとして与えておく**ということである。例文中に「社会党」という単語がある。これは「政党」という属性の値になっている。さらに、問い合わせ文中には「党首」という属性「役職」の値と「名前」という属性名がある。結局、キーワードとして、属性名「名前」と属性値「社会党」、「党首」が問い合わせ文より得られる。

これらの情報だけで問い合わせ文に答えることができる。属性名「名前」と属性値「社会党」、「党首」を入力として、

1. 入力されたすべての属性値を含むレコードを選択する。

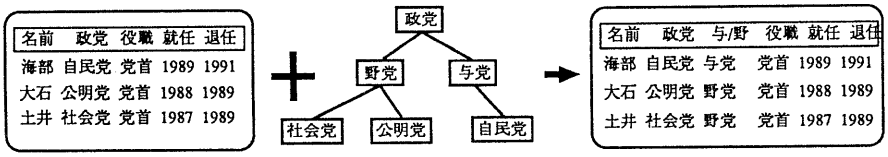

図 2: データベースの仮想化

2. 選択されたレコードの、入力された属性名の値をすべて出力する。

この場合、属性値「社会党」と「党首」を含むレコードはレコード3である。レコード3の属性名「名前」の値は「土井」である。よって、「土井」が出力される。

以上のように、入力文の解釈はキーワードベースに行なわれるので、基本アルゴリズムは非文法的な問い合わせ文に寛大である。また、一般的な辞書よりも、キーワードの情報は曖昧さや多義性が生じにくいので、基本アルゴリズムは曖昧な問い合わせ文に寛大である。さらに、言語学的な文法による解析よりも基本アルゴリズムは容易に実装できる。

以上の点から、本アルゴリズムは実用的な自然言語インタフェースの基本として有望である。SIMPLA は、この基本アルゴリズムに仮想データベースの概念を加えたモデルである。仮想データベースとは、シソーラスを用いて、対象のデータベースを仮想的に詳細化したものである (図2参照)。詳細化の方法としては、5節で詳しく述べるが、図2のような上位概念を与えるもの (「社会党」に対する「野党」) と複数の概念の合成概念を与えるもの (「就任」と「退任」に対して「在任期間」) がある。SIMPLA は、まず、問い合わせ文を仮想データベースに対する検索命令として解釈する。次に、仮想データベースに対する検索命令は実際のデータベースの検索命令に変換される。このように、仮想データベースを介して問い合わせ文を処理することによって、シソーラスと対象データベース中のデータはまったく同様に扱えるので、パーザに負担をかけることなく、シソーラスを実装できる。

3 SIMPLA の意味表現

利用者の問い合わせは、その論理的な構造においては、パターン化している。そのほとんどは、関係データベースの標準的な検索言語 SQL でいうところの、「SELECT-FROM-WHERE」型である。SIMPLA では、滅多に出現しない複雑な問い合わせを処理することは最初から諦め、多数の頻繁に使われる言いまわしの問い合わせは速く確実に解釈する方針を取ったので、問い合わせ文を「各々の属性に対する条件の宣言の連なり」と見なす。

3.1 フォームと疑似フォーム

本モデルでは、自然言語の問い合わせから検索式への変換において、2つの中間的な意味表現を経由する。それらの2つの意味表現をそれぞれ疑似フォーム、フォームと呼ぶ。

問い合わせ文 → 疑似フォーム → フォーム → 検索式

フォームとは対象データベースに対する内部的な検索式である。疑似フォームとはデータベースの詳細化されたイメージである仮想データベースに対する内部的な検索式である。

SIMPLA は、問い合わせ文を「各々の属性に対する条件の宣言と回答すべき属性の指定」と見なす。そこで、問い合わせ文の意味表現として、フォームと呼ぶ次の内部表現を用いる。フォームはユニットを要素とするリスト構造で表される。ユニットとは一つの属性に関する条件で、つぎの式で表現する:

$$r(Type, Attr, Value).$$

ここで、Type は gtr または eq または sml であり、それぞれ「より大きい」、「等しい」、「より小さい」ということを表す。上のフォームの定義中の属性名 Attr、属性値 Value はそれぞれ対象データベースのデータを表現する属性の名前とそれの取り得る値を意味するもので、次のようにしてインタフェースに予めキーワードとして与えられているとする。対象のデータベースからその属性の名前とその値との組を基本対として抽出し、ラベルベースとして保存しておく。基本対の左辺に現れる単語を属性名、右辺に現れる単語を属性値とする。

例えば、図1の政治家データベースに対して、ラベルベースとして図3を用意すれば、[r(eq, 名前, 海部)] は「名前の値が海部に等しい」という条件を表すフォームである。

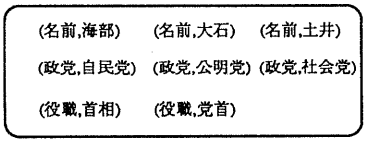

図 3: ラベルベース

上の定義から明らかなように、フォームとは検索条件の成分を並べたものであり、検索条件の成分として一つの属性に関する条件であるユニットを考えただけである。ただし、フォームは各検索条件の成分の間の論理的関係は何も表現していず、この点で検索条件そのものではない。

仮想データベースに対するフォームが疑似フォームである。疑似フォームとは疑似ユニットのリストであり、疑似ユニットとは疑似属性名 Term、疑似属性値 Walue の関係 Type を表す;

$$r(Type, Term, Walue).$$

疑似属性名、疑似属性値はそれぞれ仮想データベースの属性名、属性値である。例えば、[r(eq, 与/野, 野党)] は図4の仮想データベースに対する疑似フォームである。

名前	政党	与/野	役職	就任	退任	在任期間
海部	自民党	与党	党首	1989	1991	2
大石	公明党	野党	党首	1988	1989	1
土井	社会党	野党	党首	1987	1989	2

図 4: 仮想政治家データベース

データベースの属性名、属性値がラベルベースとして与えられたように、仮想データベースの属性名、属性値である疑似属性名、疑似属性値は語彙空間として与えられる。すなわち、仮想データベースには、疑似属性名と疑似属性値の組の集合である語彙空間が対応しており、この語彙空間はラベルベースと同様にインタフェースに予め与えられているとする。

疑似属性名と疑似属性値とは、実際は、属性名、属性値に対してマクロ表現を許したものであり、例えば、図4の仮想政治家データベースに対する疑似ユニット r(eq, 与/野, 野党) は図1の政治家データベースに対するユニットの列 [r(eq, 政党, 社会党),r(eq, 政党, 公明党)] に、問い合わせ文の解釈アルゴリズムによって、展開される。これらのマクロ表現の情報はアンカーによって与えられる (図5参照)。アンカーとは、語彙空間中の語彙対とラベルベース中の基本対との対応表である。結局、仮想データベースの実体は、語彙空間とアンカーである。例えば、図4の仮想政治家データベースに対する語彙空間とアンカーは図5のように与えられる。

以上をまとめると、図6のようになる。以下、便宜上、疑似

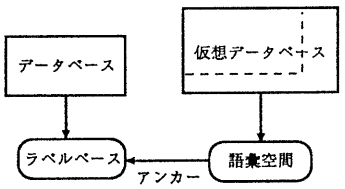

図 6: 語彙空間とアンカーによるデータベースの仮想化

属性名、疑似属性値はそれぞれ属性名、属性値を含むものとする。すなわち、語彙空間はラベルベースを含むものとする。

4 解釈アルゴリズム

図7に示すように、問い合わせ文はまず、パーザによって、仮想データベースに対する内部的な検索式である疑似フォームに変換される。次に、リアライザによって、疑似フォームは現実化されて実際のデータベースに対する内部的な検索式であるフォームに変換される。最後に、ジェネレータによって、フォームから対象のデータベースシステムの解釈する検索式が生成される。

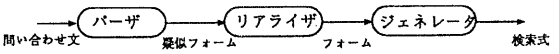

図 7: 解釈アルゴリズム

問い合わせ文を疑似フォームへ変換するのに、言語学的な文法に基づく構文解析は必要ではない。キーワードベースの解析で十分である。また、疑似フォームからフォームへの変換は、アンカーに基づいて、マクロ表現を展開することで行われる。

4.1 パーザ

問い合わせ文は次のようにして、パーザによって、語彙空間を参照することによって、疑似フォームへ変換される。

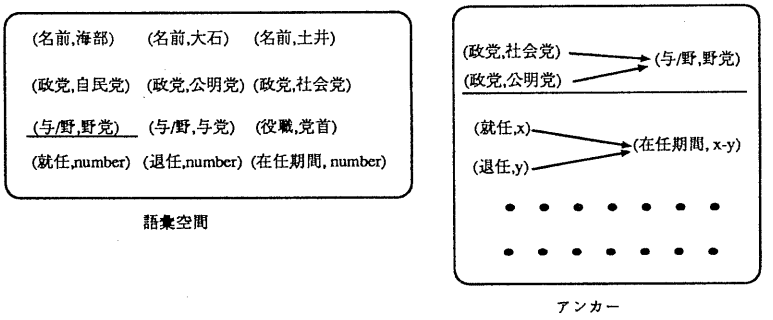

図 5: 語彙空間とアンカー

```
rl(r(Type,T,V)) --> [t(T),v(V),r(Type,R)], {(T,V) in 語彙空間}
rl(r(Type,T,V)) --> [v(V),r(Type,R),t(T)], {(T,V) in 語彙空間}
rl(r(eq,T,V)) --> [t(T),v(V)], {(T,V) in 語彙空間}
rl(r(eq,T,V)) --> [v(V),t(T)], {(T,V) in 語彙空間}
rl(r(eq,T,V)) --> [v(V)], {(T,V) in 語彙空間}
rl(r(eq,T,_)) --> [t(T)], {(T,_) in 語彙空間}
```

図 8: 簡易文法

1. 問い合わせ文から関係語、疑似属性名および疑似属性値を抽出し、リストにする。

2. 上で得たキーワードのリストに簡易文法を適用し、疑似フォームを得る。

関係語とはユニットの第1引数の Type に相当する自然言語の語彙である。

簡易文法とは、関係語、疑似属性名および疑似属性値の並びを疑似ユニットに変換するルールの集まりである。例えば、図8に示すものがある。ここで t, v および r はそれぞれ疑似属性名、疑似属性値、関係語を表し、$Type$ は eq,gtr,sml のいづれかである。例えば1行目の式は、入力列の先頭から疑似属性名 T、疑似属性値 V、タイプ $Type$ の関係語 R が続き、さらに (T, V) が語彙空間に語彙対として存在するならば、疑似ユニット $r(Type, T, V)$ を出力せよ、という意味である。

これらのルールはキーワード列に対してその先頭から繰り返し適用される。適用できたルールの数だけ疑似ユニットが出力されることになる。この疑似ユニットの集まりをリストとしたものが、求める疑似フォームである。

問い合わせ「野党の党首の名前は？」は、まず、キーワード列 [v(野党), v(党首), t(名前)] に変換され、このキーワード列に対して5行目の式を2回、6行目の式を1回順に適用することで、疑似フォーム [r(eq, 与／野, 野党), r(eq, 役職, 党首), r(eq, 名前, _)] が得られる。

4.2 リアライザ

次に、リアライザが、疑似フォームをフォームに展開する。アルゴリズムは図9のようになる。フォームへの展開は、疑似フォーム中の各疑似ユニットを、アンカーをたどることによって (図5参照)、ユニットに変換することによって行なわれる。例えば、疑似フォーム [r(eq, 与／野, 野党),r(eq, 役職, 党首),r(eq, 名前, _)] は図5のアンカーに、

$$(政党, 公明党), (政党, 社会党) \rightarrow (与／野, 野党)$$

があるので、フォーム [r(eq, 政党, 公明党),r(eq, 政党, 社会党), r(eq, 役職, 党首), r(eq, 名前, _)] に展開される。

4.3 ジェネレータ

最後にフォームは、ジェネレータによって、(対象データベースの) 検索式に変換される。アルゴリズムは図10のようになる。属性値が変数のままのユニットはその属性名を回答すべき属性名と見なし、Select 句を生成する。それ以外のユニッ

```
Algorithm  realizer(GijiForm):
Input: 疑似フォーム GijiForm
Output: フォーム Form
begin
   Form ← []
   for  each 疑似ユニット VUnit in GijiForm   do
      Form ← Form ∪ realize_one_unit(VUnit)
   retrun  Form
end

Algorithm    realize_one_unit(VUnit):
Input:  疑似ユニット VUnit
Output:  フォーム Form
begin
    F← []
    if   ∃Unit = anchor(VUnit)    then
      while   ∃Unit = anchor(VUnit) , not Unit in F    do
         F← F ∪ [Unit]
      endwhile
      Form ← realizer(F)
    else
      Form ← [VUnit]
  end
```

図 9: リアライザ

トは Where 句を生成する。問題は、各ユニットの間の論理的関係であるが、ここでは、次のヒューリステックスに基づき決定する。

1. 異なる属性名を持つユニットの関係は論理積とする。

2. 同じ属性名を持つユニットの関係は論理和とする。

例えば、先のフォーム [r(eq, 政党, 公明党),r(eq, 政党, 社会党), r(eq, 役職, 党首), print 名前] は、例えば SQL ならば、"SELECT 名前 FROM politician WHERE (政党 = 公明党 OR 政党 = 社会党) AND 役職 = 党首" に変換される。

以上のように、SIMPLA は従来の自然言語解釈の方法に較べ、非常にラフに問い合わせ文を解釈する。そのため、複雑な問い合わせ文の解釈は誤ることもある。しかし、SIMPLA は実用的な自然言語インタフェースとして、滅多に出現しない複雑な問い合わせを処理するためにシステムを複雑にするよりも、頻繁に使われる言いまわしの問い合わせを速く確実に解釈することを目的とする。

5 シソーラス

これまでは、語彙空間とアンカーは予めインタフェースに与えられたものとして、話しを進めてきた。ここでは、語彙空間とアンカーの生成について述べる。これは、シソーラスによってデータベースを仮想化することに他ならない。

SIMPLA では、シソーラスを仮想データベース中のデータとして、元もとのデータベース中のデータと平等に扱うことで、パーザの強健性を損なうことなく、シソーラスを実装する。パーザに対し、シソーラスを仮想データベース中のデータとして見せるのが、語彙空間とアンカーである。

5.1 仮想データベース

図 1の政治家データベースに対して

「野党の党首の名前は？」

という問い合わせ文を解釈するには、「野党」が属性・政党の上位概念であり、「社会党」、「公明党」その他を表すということを知っていなければならない。ところが、属性の上位概念を文法規則に持ち込めば、先の解釈アルゴリズムの簡易文法は煩雑になってしまい、パーザの強健性が損なわれる可能性がある。

そこで、SIMPLA においては、元もとのデータベースに「与野党の区別」とでもいう属性があり、そこに「野党」という属性値が書き込まれていたと考える (図 4)。すなわち、シソーラスを、仮想的に対象データベースに新たな属性を設けるかのようにしてデータとして付け加える。これによって、先の基本アルゴリズムは本質的な変更を受けることなく、シソーラスを含めた解釈を行なうことができた。

仮想データベースの実体は語彙空間とアンカーである。例えば、図 4の実体は、図 3の語彙空間とアンカーである。語彙空間の各要素は仮想データベースの疑似的な属性の名前と値のペアである語彙対であり、パーザが、ラベルベースを参照する変わりに語彙空間を参照することで、データベースは仮想化して見えるのである。

5.2 シソーラスの獲得

前節では、データベースを仮想的に拡大することによってシソーラスを実現することが、シソーラスの充実とパーザの強健性との間のトレードオフを解消するのを見た。仮想データベースの考え方は、シソーラスの獲得という点からも有意義である。すなわち、ユーザにデータベースの仮想化を指示してもらうことによって、シソーラスが獲得できるのである。

ユーザが検索命令を指示する時、データベースに直接にはない属性名や属性値が連想されていることが多い。その連想により、データベースは、人間が認識する時、変形を受け詳細化されていると考えられる。ユーザの指示に基づき、この詳細化

```
Algorithm  generator(Form):
Input:  フォーム Form
Output:  検索式 SQL
begin
    Select ← find_all_select(Form)
    Where ← find_all_where(Form)
    SQL ← generate_sql(Select, Where)
end

Algorithm  find_all_select(Form):
Input:  フォーム Form
Output:  属性名のリスト Select
begin
    Select ← []
    for each ユニット Unit in Form  do
        if  Unit = r(Type, T, V), V is uninstantiated  then
            Select ← Select ∪ [T]
    retrun  Select
end
```

```
Algorithm  find_all_where(Form):
Input:  フォーム Form
Output:  フォームのリスト Where
begin
    Where← []
    while Form\== [] do
        pop Unit=r(Type,T,V) from Form
        Or ← Formのユニットのうち属性名が
                Unitと同じTであるユニットから
                成るリスト
        Form ← それ以外のFormのユニットから
                成るリスト
        push Or into Where
    endwhile
    return Where
end
```

図 10: ジェネレータ

されたデータベースを仮想データベースとしてシステム内に実現することにより、逆にデータベースに対して行なわれたユーザの連想をとらえることができる。連想は強く領域に依存するので、この連想を表す情報によって、対象のデータベースをある程度特徴付けることができるはずである。すなわち、対象のデータベースに纏わる様々な知識、暗黙の前提を導くことができるはずである。

SIMPLA では、上のような仮想データベースの文脈において、ユーザが対象のデータベースに対して抱く語彙的な連想を、疑似フォームの使用するマクロ表現として、捉える。

図 1のデータベースはそれから連想されるような属性や値を書き込んで詳細化すると、例えば、図 4のようになる。この連想能力により詳細化されたデータベースが仮想データベースである。ユーザの指示に基づき仮想データベースをシステム内に(仮想的に)実現することにより、 ユーザが対象データベースに対して行なう連想を捉えること、すなわち疑似フォームの使用するマクロ表現を得ることができる。

実際には、つぎのオペレータ群を使用することでシステム管理者にデータベースの仮想化を指示してもらう。

属性名インデックス $term\_index(Attribute, AttributeName)$.
データベースの属性 $Attribute$ に $AttributeName$ という疑似属性名をつける。

属性値インデックス $value\_index(Attribute, Value, ValueName)$.
属性 $Attribute$ の値 $Value$ に $ValueName$ という疑似属性名をつける。

グループ化オペレータ $group(A, \{V_1, \ldots, V_n\}, A_1, V)$
ある属性 A のいくつかの属性値の集まり $\{V_1, \ldots, V_n\}$ を新しく疑似属性 A_1 の疑似属性値 V として登録する。

合成オペレータ $compound(A, \{A_1, \ldots, A_n\}, \psi)$.
属性 $A_1, \ldots, A_n$ によって新たに疑似属性 A を定義する。ここで、ψ は定義方法を与える関数である。例として、数値型の属性に対する算術操作、文字型の属性に対する結合操作などがある。

属性名インデックスと属性値インデックスにより、属性名や属性値の類義語を得ることができる。グループ化オペレータにより、属性の上位概念を、合成オペレータにより、複数の属性の合成概念を得ることができる。

例えば、図 1のデータベースに対して、図 11のようなオペレータを作用させると図 4の仮想データベースが得られる。実際には、システムはこれらのオペレータ群を解釈し、語彙空間

```
term_index(Any, Any).
value_index(Attribute, AnyTerm, AnyValue).
compound(在任期間, [退任,就任], -).
group(政党, [社会党,公明党], 与／野, 野党).
group(政党, [自民党], 与／野, 与党).
```

図 11: オペレータの使用例

とアンカーを生成し、疑似フォームの使用できるマクロ表現を得る。上の例の場合、図5の語彙空間と名前づけ関数が生成される。

6 おわりに

強健なパーザと充実したシソーラスを同時に可能とする自然言語インタフェースモデルSIMPLAを提案した。SIMPLAは、問い合わせ文の論理的な単純性を前提とし、キーワードベースのパーザを用いる。また、仮想データベースによってシソーラスを実現する。シソーラスの獲得は、データベースの仮想化の文脈で行なわれるので、自然で容易であるのを見た。

本モデルを使用することで、

- 非文法的な問い合わせ文に寛大である。
- 検索者の自由な連想に追随できる意味的な広がりをもつ。
- 回答時間が短い。
- システムの構築が容易である。

といった利点をもつ自然言語インタフェースが開発できる。実際、本モデルに基づいて、オンラインの商用データベースに対して自然言語インタフェースを開発した。

本モデルにおいては、システムとして、僅かなミスよりも多くに対する効率を重視すると述べたが、残りの僅かな場合を正しく解釈するために、Case-basedな手法を用いることを検討中である[7][8]。

参考文献

[1] Arita,S., Shimazu,H., Takashima,Y., "Simple + Robust = Pragmatic: A Natural Language Query Processing Model for Card-type Databases", Proc. of the 13th Annual Conference of the Cognitive Science Society, 1992 (to appear).

[2] 有田, 島津, 高島, "簡易型自然言語インタフェースモデル", 第5回人工知能学会全国大会予稿集, 1991.

[3] Brunner, H., "A Snapshot of Natural Language Interfaces (Panel)", Proc. of CHI-90, 1990.

[4] Grosz,B.J. et al. "TEAM: An Experiment in the Design of Transportable Natural Language Interfaces", Artificial Intelligence 32, pp.173-243,1987.

[5] Hendrix, G.G., Sacerdoti, E.D., Sagalowicz, D., and Slocum, J., "Developing a Natural Language Interface to Complex Data", In ACM Trans. on Database Systems, 1978.

[6] 牧之内顕文他 "移行性のあるデータベース自然言語インタフェース", 情処論文誌,Vol.29, No.8, pp.749-759(1988).

[7] Shimazu,H., Arita,S., Takashima,Y., "Design Tool Combining Keyword Analyzer and Case-based Parser for Developing Natural Language Database Interfaces", Proc. of COLING-92, 1992 (to appear).

[8] Shimazu,H., Takashima,Y., "Acquiring Knowledge for Natural Language Interpretation Based On Corpus Analysis", Proc. of IJCAI-91 Natural Language Learning Workshop, 1991.

[9] Tennant,H.R. et al. "自然言語を用いてコンピュータに入力する方法", 特許公報, 平2-52292.

[10] Zloof,M.M. "Query-by-Example: a Data Base Language", IBM System Journal, Vol.16, No.4, pp.324-343, 1977.