NChiql 中基于语义依存树的语言转述*)

Efficiently Paraphrasing Based on Semantic Dependency Grammar in NChiql

孟小峰　王秋月　王　珊

(中国人民大学信息学院数据与知识工程研究所　北京 100872)

Abstract　To enable natural language interfaces to databases more efficient and robust, two kinds of techniques, language paraphrasing and result analyzing, are widely used to help users to understand the system's intermediate language, and the query result meaning in some times. For the limitation of intermediate language adopted in some NLIDBs, they could not give efficient and practical paraphrasing algorithms. Based on semantic dependency tree in NChiql, we present a completed paraphrasing algorithm which is suitable to Chinese languages.

Keywords　Database query, NLIDB, Paraphrasing, Result analyzing

1. 概述

作为一个高效、鲁棒的自然语言查询接口系统(NLIDB),在进行了自然语言查询的分析之后,要有及时的反馈功能帮助用户理解系统的处理结果,避免用户对结果理解的偏差。

具体来说,NLIDB 的反馈功能,包括两个层次的含义:一是中间结果的反馈,即把系统对自然语言处理的中间结果(可以认为是系统对语言的一种理解)以某种用户可以接受的形式反馈给用户,常见的形式是把中间结果重新转换为自然语言,即转述(paraphrasing);另一种是查询结果的反馈,即对查询结果作出明确的分析和解释,帮助用户理解系统的处理结果,我们称这一部分为结果语义分析。

NChiql 在将自然语言查询转化为数据库查询语言之前,提供一套交互机制来确保系统对自然语言查询的理解和用户真正查询要求一致是极为必要的。

NChiql 系统的自然语言理解过程可分为两阶段,即首先将自然语言转化为无歧义的中间形式——语义依存树[1],然后再将语义依存树转换成为 SQL 语句。在自然语言转换成中间形式的过程中,NChiql 提供了交互手段(如学习功能)进行学习;语义依存树生成后,这表示 NChiql 系统已经理解了用户的查询语句,为了确保无误,我们应当同样提供交互手段将我们系统的理解告知用户,让用户进行确认。但由于语义依存树对用户而言不好理解,因而应当将对用户自然查询语句的理解结果用同样的自然语言方式描述出来,即转述。转述并不是原样复述,而是将理解结果告知用户,是将不规范的输入(如缩写、同义名、省略等等)替换为其标准形式,并用确切的、无歧义的标准结构来表述。

本文首先介绍 NChiql 中的中文自然语言转述的问题,提出一种适合转述的中间语言模型;其次给出转述的主要规则;并给出转述算法;最后给出相关工作比较和结论。

2. 中文自然语言查询转述方法

转述是 NChiql 系统中用户对查询语言进行纠错的主要途径。在转述时,我们应当考虑以下几个问题:

1. 歧义性问题。自然语言句子的线性结构常常引起歧义的发生。而转述的过程,正是将语义依存树的层次结构线性化的过程。因此,转述应该在清晰地表达出句子的深层语义的同时,避免重新引入歧义。

2. 句子的转述可能和原来的查询语句相同。当自然查询语句是简单句时,这种情形经常发生。在我们的系统中,允许转述句和原句相同。

语义依存树作为自然语言查询的中间形式,刻画了句子中各成分之间的相互修饰关系。自然语言理解是将线性结构层次化,构造成为语义依存树的形式,而转述是一个逆过程,即将语义依存树的层次结构扁平化(线性化)。

*)本文得到国家自然科学基金重点项目的资助(69633020)。孟小峰　博士,副教授,主要研究领域为数据库系统,嵌入与移动计算,智能查询技术等。王秋月　硕士,主要研究领域为数据库系统。王　珊　教授,博导,主要研究领域为数据库系统,数据库仓库,并行处理等。

中间形式是一个树状的层次结构，主要的节点类型有实体、属性条件和联系等，不同类型的节点保存的信息不同。以下是各类型节点的数据结构定义：

```
⟨e⟩==⟨type⟩=ENTITY
     ⟨sem⟩==
     ⟨quant⟩=ALL|SOME|MORE_THAN|
       LESS_THAN|ONLY…
     ⟨group⟩=Y|N
     ⟨modifiers⟩=⟨conds⟩

⟨conds⟩==⟨type⟩=CONDS
         ⟨c⟩=⟨cond⟩
         ⟨and⟩=⟨conds⟩

⟨cond⟩==⟨type⟩=COND
        ⟨c⟩=⟨attr⟩|⟨rel⟩
        ⟨or⟩=⟨cond⟩

⟨attr⟩==⟨type⟩=ATTR_CON
         ⟨op⟩=<|<=|>|>=|=|<>…
         ⟨exp1⟩=⟨a⟩
         ⟨exp2⟩=⟨value⟩

⟨a⟩==⟨type⟩=ATTRIBUTE
     ⟨sem⟩==
     ⟨owner⟩=⟨e⟩|⟨rel⟩
     ⟨group⟩=Y|N
     ⟨exp⟩==

⟨rel⟩==⟨type⟩=RELATIONSHIP
       ⟨sem⟩==
       ⟨pattern⟩=⟨phrase⟩
       ⟨logic⟩=NOT|ONLY|NOT_ONLY
       ⟨modifiers⟩=⟨conds⟩

⟨phrase⟩==⟨type⟩=PHRASE
          ⟨prefix⟩==
          ⟨arg⟩=⟨e⟩
          ⟨suffix⟩=⟨phrase⟩
```

说明：⟨sem⟩的取值为该节点的语义类，如⟨学生⟩、⟨课程⟩等实体类，或⟨姓名⟩、⟨成绩⟩等属性类。⟨value⟩即是一简单的数值。⟨group⟩表示是否分组。

对实体还可以使用量词⟨quant⟩：所有、一些、至少…、至多…、唯一等。

修饰联系动词结构的还可以有一些逻辑词⟨logic⟩，如：不、只、不只。

中间形式的根节点，即查询目标可以是一个ATTRIBUTE型或ENTITY型的节点，⟨a⟩中的⟨exp⟩可以是有关此节点的一个表达式，也可以是查询目标中同属一个实体或联系的一串属性，因为此阶段还未进行正式查询的处理，所以不对表达式的构成再进一步划分，而是将其作为一个整体看待。

此结构的定义中还规定了⟨conds⟩的各成分⟨c⟩(⟨cond⟩)之间是"与"的关系，而⟨cond⟩的各成分⟨c⟩(⟨attr⟩或⟨rel⟩)之间是"或"的关系，因而修饰限制实体或联系的条件实际被组织成合取式而不是析取式的形式。这样便于后面的检查结果集为空的模块⟨indirect_answer⟩的处理，因为检查每个"与"分支比检查每个"或"分支更有意义，

⟨rel⟩中的⟨pattern⟩既要描述联系的句型结构，同时又指向句型结构中出现的实体。这是使用将其划分成一个个短语片段⟨phrase⟩的方法实现的，每个片段中含有至多一个实体，⟨prefix⟩的取值为一字符串。

例如："找出供应所有A类物品给二楼出售所有B类物品的部门的供应商。"的树状结构如图1。

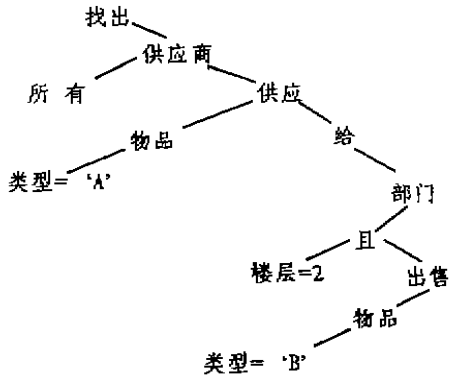

图1 转述举例

基本的转述算法是对树进行前序遍历(修饰成分在后)或后序遍历(修饰成分在前)，还要结合具体联系的谓词结构(决定变元实体的位置)。图1的树状结构线性化的结果是：

"找出供应所有类型等于'A'的物品给楼层等于2并且出售所有类型等于'B'的物品 的部门的供应商。"(后序遍历)

"找出供应商，他们供应所有物品给部门，这些物品的类型等于'A'，这些部门的楼层等于2并且出售所有物品，这些物品的类型等于'B'。"(前序遍历)

在英语中，修饰部分经常使用定语从句置于被修饰词之后，而在汉语中，修饰成分一般置前，所以后序遍历更符合汉语的语言习惯，但有时修饰中并列嵌套的成分较多时，使用并列句将一些修饰成分置后，会使语句的可读性增强。这时的语句生成策略就是前序遍历和后序遍历相结合的方式。如：

"找出供应商，他们供应所有类型等于'A'的物品给楼层等于2的部门，这些部门出售所有类型等于'B'的物品。"(前序+后序遍历)

3. NChiql中自然语言查询转述的规则

语义依存树的转述可按照以下策略来进行。

1. 确定修饰主体

即确定修饰中心词。在一条自然查询语句中，查询目标就是这条语句的中心词。所有句子的其他成分都是该中心词的修饰成分。在语义依存树中，它处于根的位置(这里我们暂且把查询词不考虑在内。)

2. 确定修饰成分的主体结构

如果修饰成分是由动词短语构成，那么修饰成分的语义结构关系便由该动词决定了，语义依存树中对动词特性的刻画是通过记录它的格来进行，格是潜存

于深层结构里的名词(包括代词)跟谓词动语之间的一种固定不变的语义结构关系。语义依存树中描述了动词的三个格成分:主格(Agent)、宾格(Object)和与格(Dative)。它们在依存树中的相互关系可用图 2 表示。

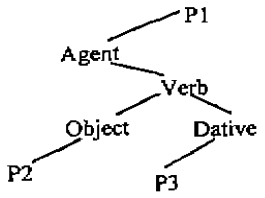

图 2 语义依存树中动词的格成分

图 2 中 P1、P2 和 P3 分别表示三个不同的短语(名词或动词短语),Verb 表示动词,那么句子 S 可表示为:

S → Verb ((P2) + Object)((P3) + Dative)
(Agent)+P1

其中'+'转化成自然语言时变为'的'。

因此,我们有算法 trans_verb,将以动词为根结点的子树转述成为一个动词语短。

算法 trans_verb

输入:动词结点 verbnode

输出:动词语短 VP。

过程:

步骤 1:输出动词名称

步骤 2:遍历动词的宾格子树 Object,先输出 Object 的孩子短语 P2,再输出 Object。

步骤 3:遍历动词的与格子树 Dative,先输出 Dative 的孩子短语 P3,再输出 Object。

3. 确定名词的修饰搭配关系

一条自然查询语句中出现的名词仅限于数据库对象名称(实体名、属性名)的范围之内。这是由自然语言接口实施于数据库系统的受限性决定的。实体名和属性名在语义依存树中必定属于下列三种情形之一:(1)父结点的修饰者。(2)被子结点所修饰。(3)动词的格。其中(3)我们在上面已经介绍过了。(1)和(2)的状态如图 3。设数据库对象 Db_object,它的父结点 Parent,子结点 Child、P1、P2 分别表示两个不同的短语(名词或动词短语),那么句子 S 可表示为:

S → ((P2) + (Child) + (Db_object) + (Dative)
(Parent)+P1

4. 确定数据库值及数据库表达式

数据库表达式是用户给出的查询条件,它表示某个属性与值的比较;而数据库值就是当比较操作为等于的数据库表达式。如"男性"、"25 岁以上"等。数据库表达式在自然查询语句中一般作为某个名词的修饰成分。设数据库表达式 Expr 及其中心词 NP,那么它们在自然语言中的表达为:Expr+NP。

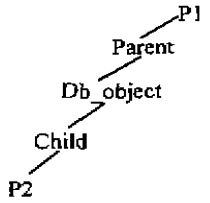

图 3 名词的修饰搭配关系

此外,就数据库表达式而言,在转述时我们一般将它表述为如下形式:

Attribute + 为/不为/大于/小于 + Value。其中 Attribute 为属性名,Value 为值。

如"男性"的转述句为"性别为男","25 岁以上"的转述句为"年龄大于 25"。从而使我们的转述方式尽量一致。

4. 转述算法

我们的转述算法 Paraphraser 具体采用后序遍历法,并综合了以上策略。如果查询目标是属性,则先生成其所属的实体或联系的短语,再显示"的"和属性表达式;如果查询目标是实体,则先生成其限定条件的短语,再显示"的"和实体名。被调用的子模块 trans(node)生成以 node 为根的子树的短语。print(str)显示 str 的内容,即将 str 拼接在最终返回结果 sentence 中。

模块·Paraphraser(tree)

输入:中间树(tree)

输出·语句(线性串 sentence)

说明·如果查询目标是属性,则先生成其所属的实体或联系的短语,再显示"的"和属性表达式;如果查询目标是实体,则先生成其限定条件的短语,再显示"的"和实体名。被调用的子模块 trans(node)生成以 node 为根的子树的短语,具体算法是后序遍历,disp(str)显示 str 的内容,即将 str 拼接在最终返回结果 sentence 中。初始时,sentence 为空。

```
paraphraser(tree)
{
 if (tree.type==ATTRIBUTE)
 {
  if (tree.owner! =NULL)
  {
   trans(tree.owner);
   disp("的");
  }
  disp(tree.exp);
 }
 else if (tree.type==ENTITY)
 {
  if (tree.quant ! =NULL) disp (tree.quant);
  if (tree.modifiers ! =NULL)
  {
    trans(tree.modifiers);
    disp("的");
  }
```

```
    disp(tree.sem);
  }
}
trans(node)
{
switch(node.type)
{
  case ATTRIBUTE:
       disp(node.exp);
                  break;
  case ATTR_CON;
       trans(node.exp1);
       disp(node.op);
       disp(node.exp2)
                  break;
  case CONDS;
       trans(node.c);
       next=node.or;
       while(next !=NULL)
       {
         disp("或者");
         trans(next.c);
         next=next.or;
       }
            break;
  case CONDS;
       trans(node.c);
       next=node.and;
       while(next !=NULL)
       {
         disp("并且");
         trans(next.c);
         next=next.and;
       }
            break;
  case ENTITY;
       if (node.quant !=NULL) disp (node.quant);
       if (node.modifiers !=NULL)
       {
         trans(node.modifiers);
         disp("的");
       }
       disp(node.sem);
            break;
  case RELATIONSHIP;
       if (node.logic !=NULL) disp (node.logic);
       next=node pattern;
       while (next !=NULL)
       {
         disp(next.prefix);
         if (next.arg !=NULL) trans(next.arg);
         next=next.suffix;
       }
       if (node.modifiers !=NULL)
       {
         disp ("的");
         trans(node.modifiers);
       }
            break;
}
}
```

5. 相关工作的比较

早期的自然语言界面系统都比较重视对中间结果自然语言的转述工作。RENDEZVOUS 系统和 PLANES 系统[5]都是使用模板(Template)的方法将系统理解出的正式查询转换成自然语句显示给用户，以供用户判定系统是否正确理解了查询请求。模板由一组固定形式的语句构成，使用时只要套用这些固定格式的句子，在句子相应的位置上填上不同词语就可以构成多个不同的完整语句。因此，为了提高系统的适用性，上述系统都必须事先预测出可能出现的不同问题从而手工构造出合适的模板。

与上面的系统不同，CO-OP 系统[3]不是使用模板方法，而是使用通用的机制直接对中间结果进行处理而生成自然语句。CO-OP 选取了元查询语言 MQL (Meta Query Language)作为中间形式，并在转述时先将其转换成树形结构；并且，CO-OP 的转述除了明确描述分析结果外，还要提请用户注意到哪些是他语句中暗含的前提，也就是将查询请求的已知部分(前提)和未知部分(陈述)分开来。NChiql 选用的方法类似于 CO-OP 系统的方法[6]，不同之处在于具体选用的中间形式不同，并且在转述中不对暗含前提进行考虑。

结束语 转述一直是数据库自然语言接口研究中的难题，在现有的 NLIDB 系统中都未能给予充分的实现。原因之一就是现有系统的中间语言不完备性。这种不完备表现在它没有融合 NLIDB 的特点，即领域知识(数据库语义)和语言知识的结合。由于 NChiql 系统的基于数据库语义的语言分析提供了语义依存树这样具有数据库语义和良好树形结构的中间语言，使得我们可以构造出上述的较为实用有效的算法，从而极大提高了系统的可用性。

参 考 文 献

1 孟小峰，刘爽，王珊. 基于数据库语义的自然语言查询分析. 计算机研究与发展，待发表

2 Kalita J K, Jones M L, McCalla G I. Summarizing Natural Language Database Responses. Computational Linguistics, 1986, 12

3 Kaplan S J. Cooperative Response from a Portable Natural Language Query System. Artificial Intelligence, 1982, 19 (2): 165～187

4 Mckeown K R. Paraphrasing Questions Using Given and New Information. American Journal of Computational Linguistics, 1986, 12

5 Waltz D L. An English Language Question Answering System for a Large Relational Database. Communications of the ACM, 1978, 21(7): 529～539

6 McKeown K R. Paraphrasing Questions Using Given and New Information. American Journal of Computational Linguistics, 1983, 9(1): 1～10

7 杰弗里·N·利奇. 语义学. 上海外语教育出版社，1987