事实库、规则库的一体化全文索引算法*)

王树西[1,2]　白　硕[1]

(中国科学院计算技术研究所软件研究室　北京 100080)[1]　(中国科学院研究生院　北京 100000)[2]

摘　要　在模式推理的计算过程中，为了快速、高效地检索到所需要的事实、规则，必须对事实库、规则库统一进行有效的组织。面对这个课题，传统的倒排索引方法已经无能为力。为此，本文给出一种新的算法，它能够对事实库、规则库统一建立一体化全文索引。在本算法的基础上，从汉语处理的实际情况出发，本文提出一种改进的算法，进一步提高了算法的效率。实验结果进一步表明，通过本算法建立的全文索引，能够快速检索到模式推理所需要的事实、规则，为模式推理工作的进行，打下了良好的基础。文章最后介绍了本算法在中文问答系统中的具体应用。

关键词　事实库，规则库，索引，模式推理，问答系统

The Integrative Index Algorithm Based on Fact-base and Rule-base

WANG Shu-Xi[1,2]　BAI Shuo[1]

(Institute of Compting Technology, The Chinese Academy of Sciences, Beijing 100080)[1]

(Graduate Student School, The Chinese Academy of Sciences, Beijing 100080)[2]

Abstract　In the process of pattern reasoning, to effectively retrieve the needed fact and rule, we must effectively organize the fact-base and rule-base. To solve this problem, traditional "inrerted index" method will be invalid. For these reasons, we propose a new algorithm which can create Integrative Index based on fact-base and rule-base. Based on this algorithm and the actual condition of chinese processing, we propose a modified algorithm which is more effectively. The experinent results indicate that this algorithm is effective and can work for the problem of pattern reasoning. In the end, the paper introduced the application of the algorithm in Question Answering System(QAS).

Keywords　Fact-base, Rule-base, Index, Pattern reasoning, Question answering system

在模式推理(Joho H,1999)的具体计算过程中，为了快速、高效地检索到所需要的事实、规则，必须对事实库、规则库进行有效的组织。面对这个课题，传统的倒排索引算法已经无能为力(王树西，2004)。为此，本文提出一种新的算法，它能够对事实库、规则库统一建立一体化全文索引。通过本算法建立的全文索引，能够快速检索到模式推理所需要的事实、规则，从而为高效进行模式推理打下了良好的基础。

1　一体化全文索引的实例

为了对全文索引有一个直观的印象，下面给出一个实例，直观地说明全文索引是如何建立的。下面以"F"表示事实，以"R"表示规则，"F1"表示事实库中的第 1 条事实，"R2"表示规则库中的第 2 条规则(Dekang Lin,2001)。

全文索引中的索引项为汉字、变量在事实、规则中的偏移量。为了计算方便，汉字、变量等同计算。例如，在事实 F1："张三和李四是夫妻"中，索引项"F1. 0"，表示汉字"张"，索引项"F1. 1"表示汉字"三"，索引项"F1. 7"表示汉字"妻"。

建立一体化索引的目的，是为了高效地进行模式推理，而模式推理的计算方法，是逆向推理。所以，在建立一体化索引的过程中，只对规则的结论建立索引，不对规则的前提建立索引。例如，在规则 R2："X 和 Y 是夫妻，X 是男的 → X 是 Y 的丈夫"中，索引项"R2. 0"表示变量"X"，索引项"R2. 1"表示汉字"是"；索引项"R2. 2"表示变量"Y"，索引项"R2. 5"表示汉字"夫"。

在计算过程中，规则中变量的表示方法采用统一的系统内部变量。例如，对于规则 R1："X 和 Y 是夫妻 → Y 和 X 是夫妻"，系统内部表示为："$1 和 $2 是夫妻 → $2 和 $1 是夫妻"。

同一个汉字可能同时出现在多条事实、规则中；同一个变量，也可能在同一条规则的结论中出现多次。所以，一个汉字的索引项可能有多个，一个变量的索引项也可能有多个。下面是建立全文索引的一个实例。

事实库原文

F1　张三和李四是夫妻

F2　李四是男的

规则库原文(输入形态，使用外部变量)

R1　X 和 Y 是夫妻 → Y 和 X 是夫妻

R2　X 和 Y 是夫妻，X 是男的 → X 是 Y 的丈夫

R3　X 是 Y 的丈夫 → Y 是 X 的妻子

规则库原文(存储形态，使用内部变量)

R1　$1 和 $2 是夫妻 → $2 和 $1 是夫妻

*)本文有关研究得到 973 项目资助，课题编号：2004CB318109。王树西　博士生，主要研究领域为计算语言学等；白　硕　博士，研究员，博士生导师，主要研究领域为人工智能、计算语言学等。

R2 $3是$4的丈夫 → $4是$3的妻子

==========

带变量的全文索引，索引项为原文中的偏移量(汉字、变量等同计算)

==========

张 F1.0

三 F1.1

和 F1.2 R1.1

李 F1.3 F2.0

四 F1.4,F2.1

是 F1.5 F2.2 R1.3 R2.1

夫 F1.6 R1.4

妻 F1.7 R1.5 R2.4

男 F2.3

的 F2.4 R2.3

子 R2.5

$1 R1.2

$2 R1.0

$3 R2.2

$4 R2.0

==========

在本实例中，多次出现一个汉字有多个索引项的情况。例如汉字“夫”，同时出现在事实 F1 和规则 R1 中，所以“夫”所对应的索引项有 2 个：F1.6、R1.4。再例如，汉字“妻”同时出现在在事实 F1、规则 R1、规则 R2 中，所以“妻”所对应的索引项有 3 个：F1.7、R1.5、R2.4。

2 一体化全文索引的结构

全文索引由两部分构成：一部分是由所有字(变量)构成链表，称为字链表；另外一个部分是每个字(变量)的偏移量分别构成的链表，称为偏移量链表。全文索引的结构如图 1 所示。

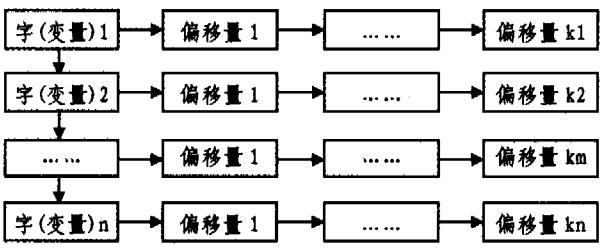

图 1 全文索引的结构

从图 1 可以看出，字链表由事实库(规则库)中所有的字(变量)构成，每个字(变量)分别对应一个偏移量链表。其中，k1，k2，…，kn ≥ 0。

① 偏移量链表的数据结构

```
struct stOffset
{
  char sType[MAX_TYPE_LEN];
  int iLineNum;
  int iOffset;
  struct stOffset * next;
};
```

在这个结构体中，sType 代表偏移量的数据类型；iLineNum 代表偏移量所在文件的行号；iOffset 代表偏移量。

② 字链表的数据结构

```
struct stLinkWord
{
  char sWord[MAX_WORD_LEN];
  struct stOffset * Offset;
  struct stLinkWord * next;
};
```

在这个结构体中，sWord 代表事实库(规则库)中的字(变量)，Offset 代表字(变量)对应的偏移量链表。

3 建立一体化全文索引的算法

依据上述数据结构，给出建立全文索引的算法：

```
void fn_vcreatewordIndexLink(string sFactBase, string sRuleBase,
struct stLinkWord * &pLinkWordHead)
{
  顺序取出事实库 sFactBase 的每一条事实 sFactLine
  {
  顺序取出事实 sFactLine 的每一个字 sWord
  {
    计算出 sWord 的偏移量 soffset;
    如果 sWord 已经在字链表 pLinkWordHead 中存在，那么，将
    soffset 插入到 sWord 的偏移量链表中;
    否则，建立一个新的结点，将 sWord 及其偏移量 soffset，插入
    到字链表 pLinkWordHead 的尾部。
  }
}顺序取出规则库 sRuleBase 每一条规则的右部 sRuleRight
{
  顺序取出 sRuleRight 每一个字(变量)sWord
  {
    计算出 sWord 的偏移量 sorrset;
    如果 sWord 已经在字链表 pLinkwordHead 中存在，那么，将
    soffsetrt 插入到 sWord 的偏移量链表中;
    否则，建立一个新结点，将 sWord 及其偏移量 soffset，插入到
    的字链表 pLinkWordHead 的尾部。
  }
}
}
```

本算法的输入参数是事实库(char * sFactBase)、规则库(char * sRuleBase)的名称，算法的输出量是字链表的头指针(struct stWord * &pHead)。

算法的时间复杂度是 $O(n^2)$。算法的主要部分是将每个字(变量)sWord 及其对应的偏移量 sOffset 插入到字链表中。

4 改进的算法

一方面，常用的汉字不是太多，一般不超过 10000 个；并且规则库中的变量个数，也不会太多，一般不会超过汉字的数量。另一方面，如果采用链表的数据结构，那么在查找所需汉字(变量)的时候，时间复杂度是 $O(n)$，效率不是太高。由于上述原因，把汉字(变量)链表的数据结构改变为递增数组的数据结构，可以进一步提高检索效率。但是应该指出，递增数组的数据结构只适用于汉语，对于英语是不适合的，因为英语单词太多。

改进之后的字(变量)数据结构是：

```
struct stWord
{
  string sWord;
  struct stOffset * Offset;
};
```

其中，sWord 代表事实库(规则库)中的字(变量)；Offset 代表字(变量)对应的偏移量链表。依据上述数据结构，得出改进之后的建立一体化全文索引的算法：

```
void fn_ vcreatWordIndexArray ( string sFactBase, string sRu ⊥
eBase, struct stWordIndex [MAX_ WORD_ NUM], int& iWord-
Num)
{
```

```
初始化,iWordNum=0;
顺序取出事实库 sFactLine 中的每一个字 sWord
{
    顺序取出事实 sFactLine 中的每一个字 sWord
    {
        计算出这 sWord 的偏移量 soffset;
        如果 sWord 已经在数组 aWordIndex 中存在,那么,将
        offset 插入到 sWord 的偏移量链表中;
        否则,将 sWord 及其偏移量 soffset,按照 sWord 递增
        的次,插入到数组 aWordIndex 中,iWordNum++。
    }
}
顺序取出规则库 sRu_eBase 每一条规则的右部 sRu_eRight
{
    顺序取出 sRu_eRight 的每一个字.(变量)sWord
    {
        计算出字 sWord 的偏移量 soffset;
        如果 sWord 已经在数组 aWordIndex 中存在,那么,将 soff-
        set 插入到 sWord 的偏移量链表中;
        否则,将 sWord 及其偏移量 soffset 按照 sWord 递增的次
        序,插入到数组 aWordIndex 中,iWordNum++
    }
}
```

本算法的输入参数是事实库(char * sFactBase)、规则库(char * sRuleBase)的名称;算法的输出量是递增排列的字(变量)数组(struct stWordVar aStrWordVar[])以及数组中字(变量)的个数(iWordVarNum)。

5 算法测试

事实库:

司马懿是司马昭的父亲

司马昭是司马炎的父亲

规则库:

X是Y的父亲,Y是Z的父亲 → X是Z的祖父

规则库(存储形态,使用内部变量)

$1是$2的父亲,$2是$3的父亲 → $1是$3的祖父

全文索引:

司 : F2.4 F2.0 F1.4 F1.0

马 : F2.5 F2.1 F1.5 F1.1

懿 : F1.2

是 : R1.1 F2.3 F1.3

昭 : F2.2 F1.6

的 : R1.3 F2.7 F1.7

父 : R1.5 F2.8 F1.8

亲 : F2.9 F1.9

炎 : F2.6

$1 : R1.0

$3 : R1.2

祖 : R1.4

根据同样的数据,通过改进的算法得到的实验结果(全文索引)是:

$1 : R1.0

$3 : R1.2

的 : R1.3 F2.7 F1.7

父 : R1.5 F2.8 F1.8

马 : F2.5 F2.1 F1.5 F1.1

亲 : F2.9 F1.9

是 : R1.1 F2.3 F1.3

司 : F2.4 F2.0 F1.4 F1.0

炎 : F2.6

昭 : F2.2 F1.6

祖 : R1.4

懿 : F1.2

测试结果表明,该算法可以有效地建立常量、变量一体化索引,为模式推理的进行打下了良好的基础。

6 在问答系统中的应用

目前的问答系统,其准确率有待提高(Zhang D, 2002; Deepak Ravichandran, 2002)。通过研究发现:之所以目前问答系统准确率不高,一个很重要的原因是目前的问答系统几乎都没有用到模式推理技术,大规模利用推理规则的问答系统极其少见。问答系统缺乏推理能力,推理系统缺乏自然语言理解能力,这是一个老问题了。正是这个问题,困扰着大型知识库系统的建设,也使花费巨大的人力、物力建立起来的知识库系统,难以面向公众开展达到一定质量的知识服务(白硕,2003)。为了解决这个问题,进行模式推理方面的工作是一个新的研究途径。而要进行模式推理,必须建立一体化全文索引,同逐个事实、规则进行匹配相比,我们建立的"一体化全文索引"大大提高了匹配的效率。简而言之,本文的工作是模式推理的基础,而模式推理,是问答系统研究的一个新途径。

结论和下一步的工作 本文提出一种建立事实库、规则库的一体化全文索引的算法,并对算法进行了改进。实验结果表明,本算法可以有效地建立一体化全文索引,从而为下一步的模式推理工作打下了良好的基础。下一步将致力于模式推理方面的工作。

参 考 文 献

1 Zhang D, Lee W S. (Singapore-MIT Alliance). Web Based Pattern Mining and Matching Approach to Question Answering. TREC, 2002

2 Ravichandran D, Hovy E. Learning surface text patterns for a question answering system. Proceedings of ACL, 2002

3 Joho H. Automatic detection of descriptive phrases for Question Answering Systems: A simple pattern matching approach: [MSc Dissertation]. Sheffield, UK: Department of Information Studies, University of Sheffield.

4 Lin Dekang, Pantel P. Discovery of Inference Rules for Question Answering. Natural Language Engineering, 2001, 7(4): 343~360

5 白硕. 大规模内容计算. 见:语言计算与基于内容的文本处理. 北京:清华大学出版社, 2003. 13~25

6 王树西. 基于自由文本的模式推理. 见:第一届全国信息检索与内容安全学术会议. 2004. 349~354