·人工智能及识别技术· 文章编号：1000-3428(2017)06-0129-07 文献标志码：A 中图分类号：TP18

基于互关联后继树的数学表达式检索

刘惠丛[1]，田冰洁[2]，田学东[1]

（1. 河北大学 计算机科学与技术学院，河北 保定 071002；2. 河北金融学院 经济贸易系，河北 保定 071051）

摘 要： 数学表达式结构复杂多样，给检索带来困难。为此，提出一种数学表达式索引与检索方法。在索引阶段，通过对 LaTeX 数学表达式特点的分析与归纳，定义面向表达式二维结构特性的数学表达式特征表示方式，将互关联后继树索引模型应用于数学表达式索引的构建，以解决树结构表示表达式的层次增长问题。在匹配阶段，设计包括精确匹配、相容匹配、子式匹配、模糊匹配等查询模式的匹配算法。在浏览器/服务器模式下采用 51 076 条数学表达式进行索引与匹配。实验结果表明，提出的方法可加快查询速度，减小索引存储空间，能够适应数学表达式的结构特点，取得较好的检索效果。

关键词： 数学表达式；索引；检索；LaTeX 格式；互关联后继树

中文引用格式： 刘惠丛，田冰洁，田学东. 基于互关联后继树的数学表达式检索[J]. 计算机工程，2017，43(6)：129-135.

英文引用格式： Liu Huicong，Tian Bingjie，Tian Xuedong. Mathematical Expression Retrieval Based on Inter-relevant Successive Tree[J]. Computer Engineering，2017，43(6)：129-135.

Mathematical Expression Retrieval Based on Inter-relevant Successive Tree

LIU Huicong[1]，TIAN Bingjie[2]，TIAN Xuedong[1]

（1. School of Computer Science and Technology，Hebei University，Baoding，Hebei 071002，China；
2. Department of Economic Trade，Hebei Finance University，Baoding，Hebei 071051，China）

【Abstract】 Aiming at the difficulties in achieving retrieval that result from the diversity of the mathematical expression structure，a method of mathematical expression indexing and retrieval is proposed. Through analysis and induction of LaTeX mathematical expression's characteristics，a mathematical expression feature representation way is defined for the two-dimensional structure characteristic in the indexing stage. And the inter-relevant successive tree indexing pattern is applied to the construction of the mathematical expression indexing，so as to solve the problem of the hierarchical growth of the tree structure representation. In the matching stage，the matching algorithm of query pattern which includes exact matching，compatible matching，sub-expression matching and fuzzy matching is designed. In the browser/server mode，51 076 mathematical expressions are used in the experiment of indexing and matching. The results show the designed indexing and retrieval method accelerates the query speed and reduces the storage space，which can adapt the structure characteristics of the mathematical expression and achieve better retrieval effect.

【Key words】 mathematical expression；indexing；retrieval；LaTeX format；inter-relevant successive tree

DOI： 10.3969/j.issn.1000-3428.2017.06.022

0 概述

数学表达式是数学信息的一种主要体现形式，由于其特殊的二维结构，针对一维结构的普通文本搜索引擎不能对其恰当处理，因此需要专门设计快速存储和搜索的方法，使搜索引擎可以检索包含表达式的数学内容。

已经出现的具备数学检索功能的方法和原型系统有 LeActiveMath[1]，MathDex[2]，EgoMath[3]，MathSearch[4]，MIaS[5] 等。其中，一些系统采用了对现有全文搜索引擎加以扩展的方法，而另外一些则专门为实现数学表达式检索而设计了索引结构。针对维基百科的 WikiMirs[6] 依据数学表达式的显性和隐性运算，利用分层泛化技术构造具有层次区分的表达式表示树并对树进行归一化，从树结构中提取表达式和表达子式构建索引。在检索阶段依据不同层次的文本匹配和结构匹配计算相似度，依据相似度对匹配结果进行排名。文献[7]同时为 MathML 建立了 Presentation 索

基金项目： 国家自然科学基金(61375075)；河北省高等学校科学技术研究重点项目(ZD2017208)。

作者简介： 刘惠丛(1989—)，女，硕士研究生，主研方向为信息检索；田冰洁，硕士；田学东(通信作者)，教授、博士、CCF 会员。

收稿日期： 2016-12-06 **修回日期：** 2017-01-27 **E-mail：** xuedong_tian@126.com

引和 Content 索引。Content 索引是对经过标准化的数学内容进行 N-grams 划分,再对数学公式抽象描述,得到抽象树倒排索引;Presentation 索引是 N-grams 倒排索引。检索阶段采用数学查询语言(Math Query Language,MQL),通过在 MathML 语言规范的基础上定义一系列元数据标签来实现查询时的通配符查询表达和组合查询表达。MathWebSearch[8] 使用置换树构建索引,置换树采用表达式的项而不是谓词。在检索阶段采用了多线程搜索技术。文献[9]也是采用置换树构建索引,引入插入偏差的概念,依据表达式基线大小修正匹配结果。在检索算法中,根据符号之间的布局查找相关表达式。文献[10]从表达式层次树中提取关键字对构造倒排索引。文献[11]利用 Trie 树构建索引,实现了对数学表达式的结构检索。文献[12]引用数学"格"的概念,提取的表达式特征构造"格",将处理后的查询表达式插入"格"中,利用"格"结构显示可视化结果。文献[13]提出一种对表达式分层索引的方法抽象出表达式的关键特征,实现了子结构查询和语义相似性查询。文献[14]把表达式在系统中存储为 MathML 代码形式,通过代码的匹配实现检索。

上述工作均从不同方面对数学表达式检索技术进行了尝试。但由于数学表达式特殊的逻辑和结构特性,设计具有较高空间和时间效率的索引结构及与之对应的匹配模型,仍是需要进一步研究的问题。针对 LaTeX 格式的数学表达式,本文通过互关联后继树索引模型[15-17],将一个数学表达式用多个互关联后继树来表示,以解决树结构表示表达式的层次增长问题,并设计相应的数学表达式索引结构和匹配算法,实现精确、相容、子式、模糊等 4 种查询模式。

1 数学表达式特征提取

采用 LaTeX 描述的数学表达式作为处理对象,基于数学公式描述结构(Formula Description Structure,FDS)[18],构造基于互关联后继树的数学表达式检索特征。

定义 1 互关联后继树的数学表达式检索特征 *EC* 为四元组(*Key*,*Level*,*Flag*,*Prestr*)。其中,*Key* 为数学表达式关键字,以添加的结束符"#"结尾;*Level* 为 *Key* 在所属表达式中的层次,0 为基准层,后续层次 *Level* 值依次递增;*Flag* 为 *Key* 在所属表达式中与相邻的上一层符号的空间位置关系,其值为 1 ~ 8 分别代表上部、上标、右部、下标、下部、内部包含、左上标和左下标关系,基准层符号的 *Flag* 值为 0;*Prestr* 表示确定 *Flag* 值时依据的符号,*Flag* 值为 0 时,*Prestr* 为 Null。

如表达式"$\frac{\sqrt{b^2-4ac}}{2a}-\frac{b}{2a}$",其 LaTeX 描述为"\[\frac{{\sqrt{{b^2} - 4ac}}}{{2a}} - \frac{b}{{2a}} \]"。获取表达式特征 *EC* 的步骤为:

1)对符号按照 *Level* 值进行升序排序。

2)在相同层次上,再对符号依据 *Flag* 升序排序。

3)合并 *Level*,*Flag*,*Prestr* 相同的符号。

所得到的数学表达式检索特征如表 1 所示。其中,"\one\frac"和"\two\frac"表示 2 个相同的字符"\frac"在表达式中出现的序号,其他同理。

表 1 表达式特征

Key	*Level*	*Flag*	*Prestr*
(\one\frac,\two-,\two\frac)	0	0	(Null)
(\sqrt)	1	1	(\one\frac)
(\two*b*)	1	1	(\two\frac)
(\two2,\two*a*)	1	5	(\one\frac)
(\three2,\three*a*)	1	5	(\two\frac)
(\one*b*,\one-,4,\one*a*,*c*)	2	6	(\sqrt)
(\one2)	3	2	(\one*b*)
(#)	0	0	

2 数学表达式索引结构

互关联后继树[15]是用于全文数据索引的一种数据结构,它只有 2 层,一层为根节点,另一层为后继节点。在将其应用于数学表达式索引时,一个数学表达式由多个互关联后继树表示,可以有效解决树结构表示表达式的层次增长问题。数学表达式是符号与符号的组合,互关联后继树使相同符号避免重复存储,相同后继的编号连续,可以加快检索速度。

利用互关联后继树建立数学表达式索引时,对运算符和运算数在互关联后继树中的节点位置和标记顺序规定如下:

1)对于双目运算符,如"+""-""×"等左右结构,运算数与运算符在同一层次,同样的标志位,运算符与运算数在同一节点,其节点信息为:对左右结构"*a* - *b* + *c*",其根节点为(*a*, -,*b*, +,*c*),后继节点为(#)。

2)对于分式运算符的上下结构,运算数在运算符的下一层次,分子和分母有不同标志位,先标记分子,再标记分母,运算符和运算数在不同节点,其节点信息如表 2 所示。

表 2 上下结构数学表达式的节点信息

上下结构表达式	根节点	后继节点
$\frac{x}{y}$	(\frac)	(*x*)
	(*x*)	(*y*)
	(*y*)	(#)

3)对于根式、求和、积分等包含结构,运算数在运算符的下一层次,上部、下部和包含运算数在不同

标志位,标记顺序为上部、下部、包含运算数,运算数和运算符在不同节点,其节点信息如表 3 所示。

表 3　包含结构数学表达式的节点信息

包含结构表达式	根节点	后继节点
$\sum_{i=1}^{n} x$	(\sum)	(n)
	(n)	($i=1$)
	($i=1$)	(x)
	(x)	(#)

4)对于角标运算符,运算数在底数的下一层次,上标和下标在不同标志位,先标记上标,再标记下标,运算数和底数在不同节点,其节点信息如表 4 所示。

表 4　角标结构数学表达式的节点信息

角标结构表达式	根节点	后继节点
a_i^2	(a)	(2)
	(2)	(i)
	(i)	(#)

当表达式的同层次中出现多种相同或不同的运算符时,标记顺序将严格按照上述规定,对某个标志位的符号统一标记完成后再进行下一标志位标记。

定义 2　符号词典存储表达式特征关键字及其键值。

为了提高检索效率,采用哈希法[19]实现关键字与键值之间的映射关系,并应用再哈希法处理冲突,使关键字分布尽可能均匀,减少 Hash 碰撞。

在关键字与其键值之间建立一个确定的对应函数关系 $Hash()$,将哈希表序列化,在符号词典中用字段 Ad 表示:

$$Ad = Hash(Key)$$

当数据库中仅含文档 Doc 时,其中的数学表达式"$\frac{\sqrt{b^2-4ac}}{2a}-\frac{b}{2a}$"符号词典如表 5 所示。

表 5　数学表达式"$\frac{\sqrt{b^2-4ac}}{2a}-\frac{b}{2a}$"的符号词典

Ad	*Key*
01e737813f253502	(\one\frac,\two-,\two\frac)
d1a81ac727b92b22	(\sqrt)
6d51589d50a35bf8	(\two$b$)
242a1deb3653caf0	(\two2,\two$a$)
7d1f15ef683cfa8c	(\three2,\three$a$)
0c09b1f7bcfd51b8	(\one$b$,\one-,4,\one$a$,c)
b108029a22097297	(\one2)
0a0c942167651c40	(#)

索引模型的基本单元是索引项,依据互关联后继树构造数学表达式索引模型,索引项包含的信息有:

1)树根(*Root*):包含符号词典的条目信息并用其 *Ad* 键值表示,还包含所提取的表达式特征中的标志位(*Flag*)和标识符(*Prestr*),即检索特征 *EC* 四元组(*Key*,*Level*,*Flag*,*Prestr*)中的 *Flag* 和 *Prestr*。

2)树叶(*Leaf*):是根节点的后继,包含符号词典的条目信息并用其 *Ad* 键值表示。

每个树叶对应的分支从 1 开始编号,分支的最大编号为其树根的度。

3)后继区间(*Interval*):如图 1 所示,当树根 $Ad(key_i)$ 下没有树叶 $Ad(key_p)$ 时,添加分支 n,对后继 $Ad(key_p)$ 编号 0,用点集 $\{0_{ip}\}$ 表示;当树根 $Ad(key_i)$ 下,存在相同的树叶 $Ad(key_j)$ 时,不再产生新的分支,使其最大编号加 1,用区间 $[Start_{ij},End_{ij}]$ 表示。

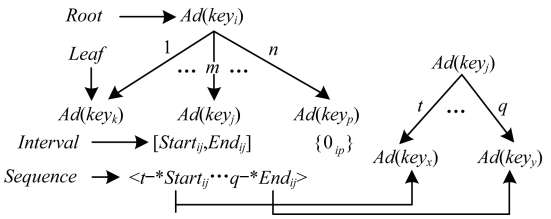

图 1　树中索引项信息

4)后继序列(*Sequence*):如图 1 所示,当树叶 $Ad(key_j)$ 为树根时,与 $Ad(key_i)$ 属于同一表达式的 $Ad(key_x)$ 在 t 分支中的编号是 $Ad(key_j)$ 的后继编号,用"分支号-编号"表示。$Ad(key_j)$ 后继编号组成的序列是后继序列,用 $<t-*Start_{ij}\cdots q-*End_{ij}>$ 表示。另外,每个表达式除#外最后一个特征关键字的后继编号为#。

5)数学表达式(*Formula*)和文档来源信息(*Source*):索引项属于哪个 LaTeX 形式的表达式和表达式的归属文档或网页链接。

基于互关联后继树的数学表达式索引构建算法如下。

算法 1　数学表达式索引的构建

输入　数学表达式特征 *EC*(*Key*,*Level*,*Flag*,*Prestr*)

输出　互关联后继树索引结构

步骤 1　读取表达式特征信息,统计相邻前驱后继关键字频率。

步骤 2　按照双字统计频度,结合数据库中相应的前驱和后继关键字的最大编号,确定当前节点的编号,进而确定树叶后继的编号。

步骤 3　将节点信息存入数据库。

步骤 4　结束。

例:数据库中仅含文档 Doc 中表达式"$\frac{\sqrt{b^2-4ac}}{2a}-\frac{b}{2a}$"时,其索引项信息如表 6 所示(*Formula* 和 *Source* 列均为"\[\frac{{\sqrt{b^2}-4ac}}{{2a}}-\frac{b}{{2a}}\]"和 Doc,表中不再赘述),互关联后继树

如图 2 所示(为了表示清楚,表中节点用关键字表示,图中节点用 *Ad* 键值表示)。

表 6 索引项信息举例

Root	*Leaf*	*Interval*	*Sequence*	*Flag*	*Prestr*
(\one\frac, \two-, \two\frac)	(\sqrt)	{0}	<1-0>	0	(Null)
(\sqrt)	(\two*b*)	{0}	<1-0>	1	(\one\frac)
(\two*b*)	(\two2, \two*a*)	{0}	<1-0>	1	(\two\frac)
(\two2, \two*a*)	(\three2, \three*a*)	{0}	<1-0>	5	(\one\frac)
(\three2, \three*a*)	(\one*b*, \one-,4, \one*a*,*c*)	{0}	<1-0>	5	(\two\frac)
(\one*b*, \one-,4, \one*a*,*c*)	(\one2)	{0}	<1-0>	6	(\sqrt)
(\one2)	(#)	{0}	<#>	2	(\one*b*)

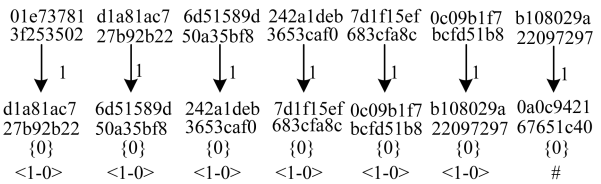

图 2 数学表达式的互关联后继树举例

基于所建立的数学表达式索引项信息所形成的数学表达式索引结构是互关联后继树中表达式信息的组织结构。索引结构由 3 个部分构成:

(符号词典,节点信息表,文档信息表)

其中,符号词典存储表达式特征关键字及其键值;节点信息表存储互关联后继树的根节点、叶子节点以及根节点的相关信息,包括根节点在表达式中的标志位和标识符特征信息和每个叶子节点对应的后继区间和后继序列;文档信息表存储完整的数学表达式以及表达式所在文档的来源信息。其逻辑结构如图 3 所示。

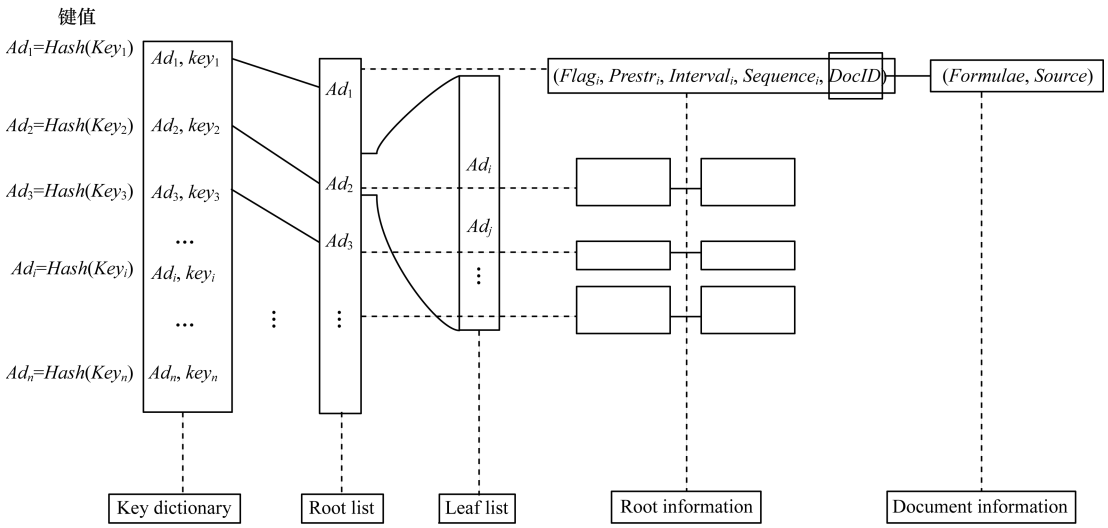

图 3 数学表达式索引结构

在图 3 中,Key dictionary 为符号词典;Root list 是树根的集合,集合中元素是 Key dictionary 中的 *Ad* 键值;Leaf list 是树叶的集合,其集合元素也是 Key dictionary 中的 *Ad* 键值;Root information 是树根信息集,与树根的集合和树叶的集合共同组成了节点信息表;Document information 是文档信息集。每个树根连接一个树叶集,每对树根、树叶对应一个树根信息,*DocID* 即文档 ID,连接根节点信息和文档信息,使索引信息分开保存又相互连接,节省存储空间。

数据库中后继树的树叶节点编号有序排列,也就是说树的所有分支从小到大编号。以数据库中仅含文档 Doc 中表达式“$\frac{\sqrt{b^2-4ac}}{2a}-\frac{b}{2a}$”,文档 Doc1 中的“$\sqrt{b^2-4ac}$”和文档 Doc2 中的“$b^2-4ac$”为例,节点编号有序的后继树如图 4 所示(此处省略符号词典和索引项信息表),其索引结构如图 5 所示。

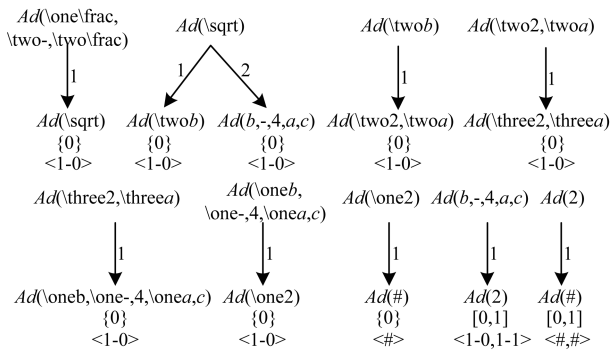

图 4 多表达式的互关联后继树举例

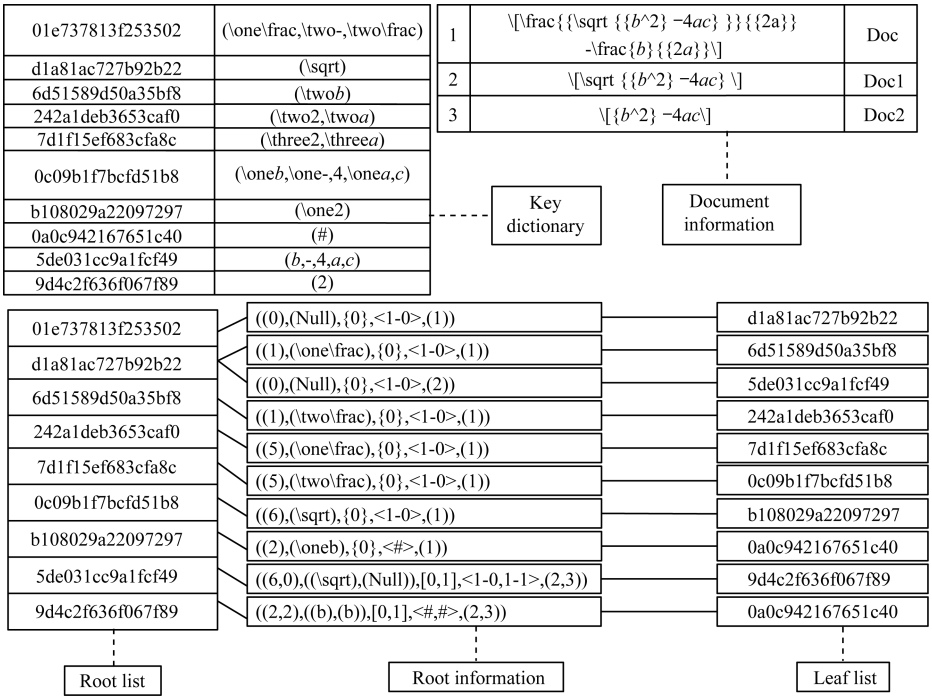

图 5　索引结构示例

3　数学表达式检索方法

以数学表达式索引结构为基础,设计了 4 种数学表达式查询模式:精确匹配,相容匹配,子式匹配,模糊匹配。

对于 3 个连续特征关键字 key_a,key_b和 key_c来说,根节点 $Ad(key_a)$,叶子节点 $Ad(key_b)$的后继序列 $<*Start_{ab},*End_{ab}>$与根节点 $Ad(key_b)$,叶子节点 $Ad(key_c)$的后继区间 $[Start_{bc},End_{bc}]$相交 $<*Start_{ab}\cdots*End_{ab}>\cap[Start_{bc},End_{bc}]$,相交结果对应的序列是以 key_a,key_b,key_c为前缀的关键字的相交序列 $<Startnext_i\cdots Endnext_i>$,相交方式如图 6 所示。

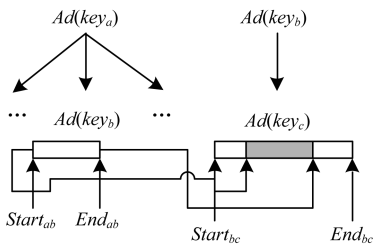

图 6　后继序列与后继区间的相交方式

1)精确匹配:查询表达式与目标表达式完全相同,对应的关键字特征也完全相同。

算法 2　数学表达式精确匹配

输入　查询表达式

输出　精确匹配结果

步骤 1　假设 n 为查询表达式的特征 EC 中 Key 的数量(#除外),$EC[i]$为第 i 个 Key 的 EC 特征($1\leqslant i\leqslant n$),$EC[i].key$,$EC[i].flag$,$EC[i].prestr$ 表示特征元素。$Root\ information(key)$为索引结构中目标表达式关键字的信息集,简写为 $RI(key)$,$RI(key).flag$,$RI(key).prestr$,$RI(key).interval$,$RI(key).sequence$,$RI(key).docID$ 表示目标表达式信息元素。

步骤 2　若 $n=1$,且$((RI(EC[1].key).prestr==\text{Null})\&\&(RI(EC[1].key).sequence==\#))\rightarrow RI(EC[1].key).docID$,则相应的 Document information 即为 Result,算法结束。

步骤 3　若 $n=2$,且$((RI(EC[1].key).prestr==\text{Null})\&\&(RI(EC[2].key).sequence==\#)\&\&(RI(EC[2].key).flag==EC[2].flag)\&\&(RI(EC[2].key).prestr==EC[2].prestr))\rightarrow RI(EC[2].key).docID$,则相应的 Document information 即为 Result,算法结束。

步骤 4　若 $n\geqslant 3$,$i=1$,$<Startnext_i\cdots Endnext_i>=(<*Start_{EC[1,2].key}\cdots*End_{EC[1,2].key}>\cap[Start_{EC[2,3].key},End_{EC[2,3].key}])\ \&\&(RI(EC[1].key).prestr==EC[1].prestr)\ \&\&(RI(EC[1].key).flag==EC[1].flag)$。

步骤 5　若($<Startnext_i\cdots Endnext_i>$为空),则 Result 为空,算法结束。

否则,若$(i=n-1)<Startnext_i\cdots Endnext_i>$中符合$(RI(EC[n].key).prestr==EC[n].prestr)\&\&(RI(EC[n].key).flag==EC[n].flag)\&\&(RI(EC[n].key).sequence==\#)$条件,相应的 Document information 即为 Result,算法结束。

否则,若$(i<n-1)$ $i++$,转到步骤 5。

步骤 6　$<Startnext_i\cdots Endnext_i>=(<Startnext_{i-1}\cdots Endnext_{i-1}>\cap[Start_{EC[i+1,i+2].key},End_{EC[i+1,i+2].key}])\ \&\&(RI(EC[i].key).prestr==EC[i].prestr)\ \&\&(RI(EC[i].key).flag==EC[i].flag)(i>1)$,转到步骤 5。

当查询表达式的特征 *EC* 中 *key* 的数量 $n \leqslant 2$(#除外)时,算法在 $O(1)$的时间内结束;当 $n > 2$ 时,步骤 4 中的求交运算复杂度是 k_1,即序列 $< Startnext_1 \cdots Endnext_1 >$ 中元素的个数,步骤 6 的运算复杂度为 k_i,所以在查询表达式有解的情况下该算法的复杂度是 $O(k_1 + k_2 + \cdots + k_{n1})$,无解的情况下,复杂度为 $O(k_1 + k_2 + \cdots + k_j)$($j$ 表示求交后无解的情况)。因为随着相交次数的增多,相交结果越来越少,所以之后的 k_i $(1 < k < n - 1)$都小于 k_1,总的复杂度 $O(k_1 + k_2 + \cdots + k_{n1})$低于 $O(k_1 \times n)$,即远低于 $O(n)$。

2)相容匹配:查询表达式是目标表达式的子集,即查询表达式能够与目标表达式的一部分精确匹配。在算法 2 中,不限制第一个关键字的 *Prestr* 特征和最后一个关键字的 *Sequence* 特征,得到的文档信息即为相容匹配结果。

3)子式匹配:与相容匹配情况相反,目标表达式是查询表达式的子结构,即查询表达式包含整个完整的目标表达式。此种匹配方法是在提取的查询表达式特征基础上,提取出查询表达式子结构特征,并对其不含有符号编号的子结构特征进行精确匹配,得到的文档信息即为子式匹配结果。

4)模糊匹配:目标表达式与查询表达式在一定程度上匹配,目标表达式包含查询表达式子结构,关键字特征不必相同,公式可拆分,即目标表达式与查询表达式部分相同。对查询表达式的子结构特征进行相容匹配,得到的文档信息即为模糊匹配结果。

4 实验与结果分析

在浏览器/服务器模式下实现了数学表达式检索的原型系统,服务器端硬件环境为四核 2.3 GHz 中央处理器和 8 GB 内存;系统环境为 64 位 Windows Server 2012 操作系统和 SQL Server 2012 数据库系统。客户端硬件环境为双核 2.5 GHz 中央处理器和 8 GB 内存;系统环境为 64 位 Windows 7 操作系统。

实验数据是 51 076 条采集自维基百科中的数学表达式。对于查询公式“$\frac{\sqrt{b^2-4ac}}{2a} - \frac{b}{2a}$”的检索结果有:$\frac{\sqrt{b^2-4ac}}{2a} - \frac{b}{2a}$,$\sqrt{b^2-4ac}$,$b^2-4ac$,$b-4ac$,$\int \frac{\sqrt{b^2-4ac}}{2a} - \frac{b}{2a}\mathrm{d}x$,$\int \sqrt{b^2-4ac}\,\mathrm{d}x$,$\sqrt{b-4ac}$,…。其中“$\frac{\sqrt{b^2-4ac}}{2a} - \frac{b}{2a}$”为精确匹配;“$\sqrt{b^2-4ac}$”“$b^2-4ac$”和“$b-4ac$”为子式匹配;“$\int \frac{\sqrt{b^2-4ac}}{2a} - \frac{b}{2a}\mathrm{d}x$”为相容匹配;其他为模糊匹配。

本文与文献[20]的索引文件大小情况分别如表 7、表 8 所示。可见随着公式数量的增加,本文方法的索引文件大小持续增长,但增长幅度不大,这是因为表达式中相同节点不会重复存储。

表 7 本文方法实验数据

表达式数量	索引文件大小/MB
10 276	1.99
20 077	4.15
40 165	8.57
51 076	11.19

表 8 文献[20]方法实验数据

表达式数量	索引文件大小/MB
1 036	1.39
10 096	13.30
100 048	130.00

随机抽取 418 条公式进行检索效率实验,结果如表 9 所示。文献[20]方法的检索效率如表 10 所示。从表 9 中可以看出,本文方法的检索时间随着公式数量的增长而增长,这是因为随着解析的公式数量的增长,后继区间和后继序列中数据增多,其后继序列与后继区间相交时间随之增长。实验数据表明,本文方法的效率在可接受范围内。

表 9 本文方法检索效率 s

待检索表达式数量	平均检索时间
10 276	0.163
20 077	0.372
40 165	0.522
51 076	0.587

表 10 文献[20]方法检索效率 s

待检索表达式数量	最长检索时间
1 036	0.147
10 096	0.431
100 048	0.532

5 结束语

本文提出一种数学表达式索引与检索方法,依据互关联后继树模型构造了适用于数学表达式的索引与检索模型,模型中聚集相同后继节点,使后继节点编号有序,加快了查询速度,减小了索引存储空间。通过分析表达式特点,在检索阶段设计了 4 种查询模式,丰富了查询结果。通过大量数据实验验证了该方法的可行性和有效性。

下一步的工作是针对数学表达式的二维复杂特性,加强对表达式特征的分析,减少树结构的横向增长,并根据用户需求的不同,在进一步的实验中补充同层次部分匹配的相关表达式检索,使系统可以检索出更符合使用者查询的结果,满足更多用户的需求。

参考文献

[1] Libbrecht P, Melis E. Methods to Access and Retrieve Mathematical Content in Active Math[C]//Proceedings of International Congress on Mathematical Software. Berlin, Germany: Springer, 2006: 331-342.

[2] Miner R, Munavalli R. An Approach to Mathematical Search Through Query Formulation and Data Normalization [C]//Proceedings of International Workshop on Mathematical Knowledge Management. Berlin, Germany: Springer, 2007: 342-355.

[3] Mišutka J, Galamboš L. System Description: EgoMath2 as a Tool for Mathematical Searching on Wikipedia. org[C]// Proceedings of International Conference on Intelligent Computer Mathematic. Berlin, Germany: Springer, 2011: 307-309.

[4] 刘志伟. 数学搜索引擎研究[D]. 兰州:兰州大学, 2011.

[5] Sojka P, Liška M. Indexing and Searching Mathematics in Digital Libraries [C]//Proceedings of International Conference on Intelligent Computer Mathematics. Berlin, Germany: Springer, 2011: 228-243.

[6] Hu Xuan, Gao Liangcai, Lin Xiaoyan, et al. WikiMirs: A Mathematical Information Retrieval System for Wikipedia [C]// Proceedings of the 13th ACM/IEEE-CS Joint Conference on Digital Libraries. New York, USA: ACM Press, 2013: 11-20.

[7] 景　珂. 网络数学搜索中的数学查询语言与索引的研究[D]. 兰州:兰州大学, 2009.

[8] Kohlhase M, Anca S, Jucovschi C, et al. MathWebSearch 0.4, A Semantic Search Engine for Mathematics[EB/OL]. (2010-12-22). http://mathweb. org/projects/mws/pubs/mkm08. pdf.

[9] Schellenberg T, Yuan Bo, Zanibbi R. Layout-based Substitution Tree Indexing and Retrieval for Mathematical Expressions [C]//Proceedings of Society of Photo-optical Instrumentation Engineers Conference. Washington D. C., USA: IEEE Press, 2012: 263-271.

[10] Stalnaker D. Math Expression Retrieval Using Symbol Pairs in Layout Trees [D]. Rochester, USA: Rochester Institute of Technology, 2013.

[11] 孙　净. 基于 Trie 树的数学表达式运算结构检索[D]. 保定:河北大学, 2015.

[12] Nguyen T T, Hui Siu-cheung, Chang Kuiyu. A Lattice-based Approach for Mathematical Search Using Formal Concept Analysis [J]. Expert Systems with Applications, 2012, 39(5): 5820-5828.

[13] 卢　托. 科技文档中数学公式的描述与索引[D]. 武汉:华中科技大学, 2007.

[14] 刘东阁. 基于 MathML 的公式检索系统的设计与实现[D]. 沈阳:东北大学, 2009.

[15] 申　展, 王建会, 吴爱民, 等. 互关联后继树模型——一种新颖的全文搜索模型[J]. 计算机科学, 2003, 30(10): 351-354.

[16] 申　展, 江宝林, 张　谧, 等. 互关联后继树模型及其实现[J]. 计算机应用与软件, 2005, 22(3): 7-9.

[17] Yang Chuanyao, Li Yuqin, Wang Zhenghua, et al. A Yellow Page Information Retrieval System Based on Sorted Duality Inter-relevant Successive Tree and Industry Ontology[C]// Proceedings of the 8th Inter-national Conference on Software Engineering. Washington D. C., USA: IEEE Press, 2007: 1147-1152.

[18] Tian Xuedong, Yang Songqiang, Li Xinfu, et al. An Indexing Method of Mathematical Expression Retrieval[C]// Proceedings of the 3rd International Conference on Computer Science and Network Technology. Washington D.C., USA: IEEE Press, 2013: 574-578.

[19] 严蔚敏, 吴伟民. 数据结构(C 语言版)[M]. 北京:清华大学出版社, 2009.

[20] Lin Xiaoyan, Gao Liangcai, Hu Xuan, et al. A Mathematics Retrieval System for Formulae in Layout Presentations[C]//Proceedings of the 37th International Conference on Research and Development in Information Retrieval. New York, USA: ACM Press, 2014: 697-706.

编辑　顾逸斐

(上接第 128 页)

参考文献

[1] 潘　超, 杨良怀, 龚卫华, 等. 模式匹配研究进展[J]. 计算机系统应用, 2010, 19(11): 265-275.

[2] 杜小坤, 李国徽, 王江晴, 等. 基于信息元的模式匹配方法[J]. 软件学报, 2015, 26(10): 2597-2613.

[3] 刘卫国, 胡勇刚. DHSWM:一种改进的 WM 多模式匹配算法[J]. 中南大学学报(自然科学版), 2011, 42(12): 3766-3767.

[4] 嵩　天, 李冬妮, 汪东升, 等. 存储有效的多模式匹配算法和体系结构[J]. 软件学报, 2013, 24(7): 1651-1665.

[5] Zhong Qiuxi, Wan Hui, Xie Peidai, et al. An Efficient Packet Pre-filtering Algorithm for NIDS [J]. Lecture Notes in Electrical Engineering, 2012, 126: 113-120.

[6] Prabha K, Sukumaran S. Single-keyword Pattern Matching Algorithms for Network Intrusion Detection System [J]. International Journal of Computer and Internet Security, 2013, 5(1): 11-18.

[7] 姜庆民, 吴　宁, 刘伟华. 面向入侵检测系统的模式匹配算法研究[J]. 西安交通大学学报, 2009, 43(2): 59-60.

[8] Kalita N, Sharma R, Borah S. eKMP: A Proposed Enhancement of KMP Algorithm [J]. Computational Intelligence in Data Mining, 2015, 3: 479-487.

[9] 朱保锋, 宋　艳. 一种改进的 BM 算法性能分析[J]. 中州大学学报, 2015(3): 114-116.

[10] 钱松波, 刘嘉勇. 一种适于 HTTP 数据还原的 QS 改进算法[J]. 通信技术, 2015(3): 351-355.

[11] 马占飞, 杨树英, 郭广丰. 一种快速的基于 BM 模式匹配的改进算法[J]. 控制与决策, 2013, 28(12): 1857-1858.

[12] 金　凌. 面向比特流的未知帧头识别技术研究[D]. 上海:上海交通大学, 2011.

[13] 巫喜红. 改进的 QS 模式匹配算法的性能分析[J]. 计算机工程与应用, 2014, 50(2): 44-48.

[14] 曾传璜, 段智宏. 一种改进的 QS 串匹配算法[J]. 计算机与数字工程, 2010, 38(7): 48-49.

[15] 王和洲. 面向比特流的链路协议识别与分析技术[D]. 合肥:中国科学技术大学, 2014.

编辑　顾逸斐