文章编号:1001 - 9081(2009)03 - 0854 - 04

一种改进的 XML 查询重写算法研究

李 锐,吴开贵
(重庆大学 计算机学院,重庆 400044)
(leasye@ gmail. com)

摘 要:查询重写是数据集成的一个关键问题,它是将用户的查询请求自动重写为直接面向数据源的查询请求。最近 Michigan 大学和 IBM 的 AImaden 研究中心提出了一种新的基于约束的 XML 查询重写算法,但是该算法没有考虑复杂模式匹配重写问题,使得该算法应用受到限制。在原来的算法重写思想基础上,提出了一种改进的 XML 查询重写算法,扩大原算法的应用范围,并分析了改进算法的正确性和时间复杂度。

关键词:XML;查询重写;模式匹配;数据集成
中图分类号: TP311. 133. 1 **文献标志码**:A

Extended algorithm for XML query reformulation

LI Rui, WU Kai-gui
(*College of Computer Science, Chongqing University, Chongqing* 400044, *China*)

Abstract: Query reformulation is a key issue of the data integration, and it automatically rewrites the user's query request to the data source request directly. Recently the University of Michigan and the IBM AImaden Research Center presented a novel algorithm for constraint-based XML query reformulation, but it did not consider the problem of complex schema matching and limited the algorithm application. Based on the original query reformulation algorithm, an extended algorithm for XML query reformulation was presented, expanding the application scope of the original algorithm. The correctness and time complexity of the extended algorithm were also given.

Key words: XML; query reformulation; schema matching; data integration

0 引言

数据集成系统为多个数据源提供了统一的访问接口[1],用户基于中介模式(或称全局模式 ,Mediated Schema)提交一个查询,数据集成系统通过源模式与中介模式的映射关系将该查询重写为面向各个数据源的子查询。查询重写是实现这个重写过程的关键。查询重写依赖源模式与全局模式之间的模式映射,构建这种模式映射关系的基本方法主要有三种:GAV[2](Global-As-View), LAV[3](Local-As-View)和 GLAV(Global-Local-As-View)[4]。GAV 指全局模式作为数据源模式上的视图被定义,如 TSIMMIS[5]等;LAV 指数据源模式作为全局模式上的视图被定义,如 Information Manifold[6]等;GLAV 是 GAV 和 LAV 结合的产物,由全局模式视图和源模式视图结合而成。在本文中,我们采用 GLAV 作为模式间映射方法,它结合了 GAV 和 LAV 的优势,能够提供更具表达能力的模式映射。

XML 是一种半结构化的数据模型,它的许多特性[7]使得它可以描述不规则的数据,能够集成来自不同数据源的数据。文献[8]中提出了基于约束的 XML 查询重写算法,但是该算法不能解决复杂模式匹配问题。

例 1:假设有某个应用需求收集论文资料,并且根据需求已经建立了中介模式 med,如图 1 所示,其中符号 Title、Author、Abstract、Txt 分别代表"标题"、"作者"、"摘要"和"内容",符号 T、K 则分别起到了 XML 模式中外部引用的作用。图 2 为数据源模式和全局模式的映射集合,模式映射 M_1 和 M_2 说明了 src_1 和 src_2 中的论文信息元组是如何映射到中介模式中相应的元素。图 3 为中介模式的约束。假设存在查询 Q:

```
Q ::= for p in med. PaperCollection, t in med. Texts
      where p. Paper. K = t. Text. K
      return [Title = p. Paper. Title, Author = p. Paper. Author,
      Abstract = p. Paper. Abstract, Text = t. Text. Txt]
```

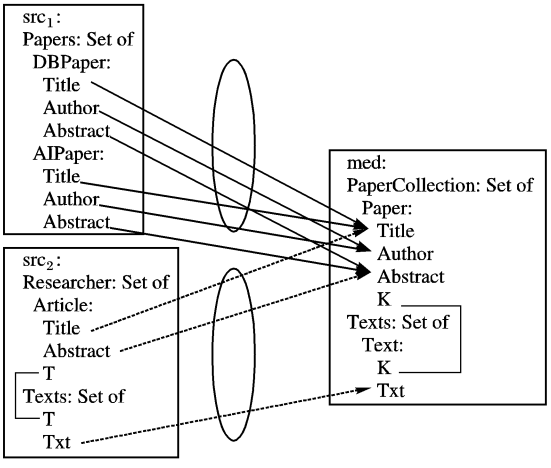

图 1 数据源和中间层模式

Q 为查询学者姓名,发表论文的标题、摘要及论文内容。根据文献[8],算法根据给定的数据源模式和中介模式以及数据源模式和中介模式映射集合,构造一系列映射规则集合

收稿日期:2008 - 09 - 12;**修回日期**:2008 - 11 - 10。

作者简介:李锐(1982 -),男,湖北人,硕士研究生,主要研究方向:信息集成; 吴开贵(1966 -),男,重庆人,副教授,主要研究方向:信息与网络安全、计算机网络。

Mrs。构造映射规则集合是该算法关键的一步,它是转换和分解全局查询到数据源查询的基础,但是对每个映射中的 exists 子句中的表达式构造一个映射规则具有局限性,当中介模式和某个数据源模式的模式匹配基数不是 1∶1 时,文献[8]中的算法就不适用。

所谓模式匹配基数就是限定待匹配的两个模式中元素参与度的规则,包括 1∶1,1∶n,n∶1,m∶n,如上面映射 M_1 的中介模式中的元素 p′. Paper. Title,p′. Paper. Author 和 p′. Paper. Abstract 在模式 src_1 中分别对应多个元素。m∶n 的复杂匹配,因涉及到较多的模式语义约束,很难完全自动实现,本文只探讨基数为 1∶1,1∶n,n∶1 的匹配问题。

```
M1:
foreach p in scr1.Papers
exists p' in med.PaperCollection, t' in med.Texts
where p'.Paper.K=t'.Text.K
with p'.Paper.Title=
[p.DBPaper.Title, p.AIPaper.Title]
and p'.Paper.Author=
[p.DBPaper.Author, p.AIPaper.Author]
and p'.Paper.Abstract=
[p.DBPaper.Abstract, p.AIPaper.Abstract]
M2:
foreach r in scr2.Researcher, t in src2.Texts
where r.Article.T=t.T
exists p' in med.PaperCollection, t' in med.Texts
where p'.Paper.K=t'.Text.K
with p'.Paper.Title=[r.Article.Title]
and p'.Paper.Abstract=[r.Article.Abstract]
and t'.Text.Tex=[t.Tex]
```

图 2　映射集合

```
C1:
 for p1 in med.PaperCollection,
     p2 in med.PaperCollection,
    [p1.Paper.Title=p2.Paper.Title and
     p1.Paper.Author=p2.Paper.Author
     →p1.Paper.K=p2.Paper.K]
C2:
 for t1 in med.Texts, t2 in med.Texts
  [t1.Text.K=t2.Text.K
  →t1.Text.Txt=t2.Text.Txt]
```

图 3　中介模式的约束

1　相关知识介绍

1.1　查询重写算法

重写算法可分为基于传统关系模型的重写算法和基于半结构化视图和数据模型的重写算法。关系查询重写算法又分为 Bucket 算法(桶算法)、Inverse-Rules 算法(逆向规则算法)和 MiniCon 算法;半结构化重写算法又分为 MSL(Mediator Specification Language)算法和 COMMIX(content-oriented massive information integration based on XML)算法。我们采用的 XML 查询重写算法是在文献[8]算法的基础上进行改进以支持复杂模式匹配问题。文献[8]算法本质上利用了关系查询重写中的逆向规则算法[9],通过利用映射规则将中介模式的每个元素用数据源中元素表示从而得到映射规则,利用这些映射规则对原有查询进行替换以达到重写的目的。

1.2　查询包含和查询相等

对于查询 Q_1,Q_2 和数据库 D,查询 Q_1 包含在查询 Q_2 中,记作 $Q_1 \subseteq Q_2$,其含义是 Q_1 对于任何数据库实例 D 的查询结果都是 Q_2 查询结果的子集,即 $Q_1(D) \subseteq Q_2(D)$。如果 $Q_1 \subseteq Q_2$ 并且 $Q_2 \subseteq Q_1$,则称为 Q_1 和 Q_2 查询相等,记作 $Q_1 \equiv Q_2$。

1.3　最大查询重写和等价查询重写

设 Q 为一个查询,而 $V = V_1, V_2, \cdots, V_n (n \geqslant 1)$ 为一组视图定义的集合。L 是一个查询语言,如果 Q' 是 L 上的查询并且仅仅引用 V 上的视图同时 $Q' \subseteq Q$,并且不存在 L 上仅仅引用 V 上视图的查询 Q_1,使得 $Q' \subseteq Q_1 \subseteq Q$ (Q' 不等价于 $Q1$),则称对于查询语言 L,Q' 是 Q 的最大查询重写,如果 $Q' \equiv Q$ 则称 Q' 是查询 Q 的等价重写。

1.4　查询语言和查询规则

本文中的查询语言和查询规则具有如下形式(见图 4)。在查询语言中,每项 x_i in g_i 为一个表达式,x_i 为原子类型,g_i 为 SetOf 类型。B 为一组介于 $x_i, \cdots, x_n$ 之间起联接作用的等式。r 为查询结果,其形式可以为记录元组,如 $[A_1 = r_1, \cdots, A_k = r_k]$,$A_i$ 称作标签(label),r_i 可以为原子类型、记录类型和嵌套的子查询。映射语言中表达式 $x_i(g_i)$ in $g_i(g'_i)$,变量 $x_i(y_i)$,两个 where(B_1 和 B_2)子句都和查询语言的定义相同。最后,在 with 子句 $e'_i = [e_{i1}, \cdots, e_{im}]$ 中表示中介模式和数据源模式的映射关系,它们的模式匹配基数可能为 1∶1 或 1∶n。

1.5　模式匹配

模式匹配是一个涉及知识表示方法、机器学习、信息检索等领域的复杂问题,匹配过程通常结合一定的启发式规则,一般没有严格的数学理论支持。它一般定义为形如 Matching(I_1, I_2, A) 的函数,$I_1 \in \{ < S_1, D_1 > | S_1$ 为模式信息,D_1 为满足模式 S_1 的数据实例$\}$,I_2 的定义和 I_1 相同。A 为匹配过程中需要利用的辅助信息(包括字典、人的判断信息等)。本文通过设计一种数据结构,同时在这个数据结构上定义一列操作来解决映射语言中 with 子句中存在的复杂匹配问题。

```
q::=for x1 in g1, ···, xn in gn
    where B
    return r
```

(a) 查询语言

```
M::=foreach x1 in g1, ···, xn in gn
    where B1
    exists y1 in g1', ···, ym in g'm
    where B2
    with e1 =[e11, ···, e1m] and ··· and
         ek =[ek1, ···, ekm]
```

(b) 映射语言

图 4　查询语言和映射语言

2　改进的 XML 查询重写算法的主要思想

为了解决由模式映射集合到构造一系列映射规则集合 Mrs 过程中出现的复杂匹配问题,我们构造一个数据结构和对它进行的操作。改进算法的主要思想是分析中介模式与数据源模式的映射,如果映射中出现复杂匹配,我们需要借助辅助信息来使那些元素之间的匹配组合是合理的,这样我们就可以为每个映射中的 exists 子句的每个表达式构造多个映射规则。

定义 1　模式映射。对于数据源模式 S,中介模式 D,我们把模式之间的映射抽象为一个表达式为:

$$\{D, P_i\} \rightarrow C^m \{S, P_e^+\} \tag{1}$$

其中 P_i 为 D 中任意元素的路径表达式,P_e^+ 为与 P_i 对应的多个元素路径表达式,C^m 为模式匹配基数如 1∶1,1∶n,n∶1。

定义 2　匹配辅助信息。给定一棵数据源模式树,设 r 是其根节点,A_i 是其一节点,并有父节点 A,那么从 r 至 A 的路径成为节点 A_i 的依赖路径。把表达式(1)分解成一系列 1∶1 简单映射表达式为:

$\{D, P_i\} \rightarrow \{S, P_e\}$ (2)

在 P_e 的依赖路径上,寻找关于 P_e 的路径前缀 P'_e,P'_e 即为模式匹配的辅助信息。

定义 3 数据源元素数据结构。我们为数据源中的每个原子元素定义一个与之关联的数据结构,如图 5 所示。

ID	SrcName	EName	SInformation	PJoin_ID	Join_ID

图 5 数据结构

ID 为元素的标识,SrcName 为数据源的名字,EName 为元素的名字,SInformation 为辅助信息,PJoin_ID 和 NJoin_ID 为可以和该元素组合的前一个元素 ID 和后一个元素 ID。

定义 4 组合判断。在 with 子句中元素之间的映射不为 1:1 时需进行模式匹配判断。设有元素 e_1 和 e_2,与之关联的数据结构为 E_1 和 E_2,e_1 和 e_2 是否可以组合满足下面两个规则:

R_1:E_1. SrcName 和 E_2. SrcName 相同。

R_2:E_1. SInformation 即 e_1 的路径前缀和 E_2. SInformation 即 e_2 的路径前缀属于同一条路径(包括 E_1. SInformation 或 E_2. SInformation 为空的情况)。

规则 R_1 限制了不同数据源的属性之间相互组合的可能,规则 R_2 保证了对应同一个实体的不同属性的相互组合,限制了对应不同实体的属性之间相互组合的可能。

如果不满足规则 R_1 或者 R_2,则表明 e_1 和 e_2 不能进行组合,当同时满足这两个规则后,e_1 和 e_2 能进行组合,同时更改数据结构内容使 E_1. NJoin_ID = E_2. ID(E_2. PJoin_ID = E_1. ID)或 E_2. NJoin_ID = E_1. ID (E_1. PJoin_ID = E_2. ID)。

3 主要算法的实现和分析

3.1 算法的实现

第 1 步 推导生成数据源数据结构集合 D

输入:数据源模式定义集合 S = {S_1, S_2, …, S_n};

输出:数据源数据结构集合 D = {D_1, D_2, …, D_n};

1)根据数据源模式初始化数据源中每个原子元素的数据结构。

设 S_i 为第 i 个数据源模式,d_{ij}为数据源中的第 j 个元素,对应的数据结构为 D_{ij},根据 d_{ij}在数据源中的信息初始化 D_{ij},这时 D_{ij}. PJoin_ID = NULL, D_{ij}. NJoin_ID = NULL, $D_i = D_{i1} \cup D_{i2} \cup \cdots \cup D_{im}$ ($m \geq 1$)。

2)判断数据源元素相互组合的可能。

搜索数据源模式 S_i 下所有原子元素的数据结构,根据它们的匹配辅助信息,依据定义 4 来判断它们能否相互组合,并决定 PJoin_ID 和 NJoin_ID 的值,同时一个 ID 只能和一个 PJoin_ID 或 NJoin_ID 关联,这样保证相互组合元素的完备性和唯一性。设 src_1. Papers. DBPaper 下的元素 Title, Author, Abstract 的数据结构分别为 DBTi, DBAu 和 DBAb 我们得到 DBTi. PJoin_ID = NULL, DBTi. NJoin_ID = DBAu. ID, DBAu. PJoin_ID = DBTi. ID, DBAu. NJoin_ID = DBAb. ID, DBAb. PJoin_ID = DBAu. ID, DBAb. NJoin_ID = NULL。分析某个元素的数据结构信息,通过 PJoin_ID, NJoin_ID 来获取一系列和该元素能进行组合的元素集合。

第 2 步 推导生成映射规则集合 M_{rs}

输入:中介模式定义 med;数据源模式定义集合 S;数据源模式和中介模式的映射集合,如 $Mapping_1$, $Mapping_2$, …;数据源数据结构集合 D。

输出:映射规则集合 M_{rs}。

1)首先为全局模式中的每个根产生一个规则。

2)再查看给定的映射集合,判断给定模式之间的模式匹配基数,如果为 1: n 时,这样我们为给定映射中的 exists 子句的每个表达式构造 n 个映射集合,其中,映射中的 foreach 子句以及和它关联的 where 子句变为映射规则中查询的 for 子句和 where 子句,这时判断 with 子句中数据源元素之间的相互组合性,把能相互组合在一起的元素构成一个集合,用来填充 return 子句中原子元素的值,对于集合类型,则用一个 skolem 函数产生和它相关联的 SetID,这时转 4);否则转 3)。

3)根据给定的模式映射集合,对每个映射中的 exists 子句的每个表达式构造一个映射规则,对 foreach 子句及和它关联的 where 子句、集合类型、处理方式同过程 2 相同,对于 with 子句我们用源表达式来填充 return 子句的原子元素的值。

4)然后对得到的所有映射规则进行迭代,组合所有的在规则头中具有相同的 Skolem 函数的规则,为全局中的每个集合类型获得一个单独的定义。例 1 产生的映射规则集合如图 6 所示。

```
Rule0 : med=[PaperCollection=SKLM med.0(), Texts=SKLMmed.1()]
Rule13 : SKLMmed.0 ()= λ ().{
 for p in src1.Papers
 Return[Paper=[Title=p.DBPaper.Title,
  Author=p.DBPaper.Author,
  Abstract=p.DBPaper.Abstract,
  K=SKLM125 (p.DBPaper.Title, p.DBPaper.Author,
          p.DBPaper.Abstrct)]]
  ∪for r in src1.Papers
  return[Paper=[Title=p.AIPaper.Title,
       Author=p.AIPaper.Author, Abstract=p.AIPaper.Abstract,
       K=SKLM126(p.AIPaper.Title, p.AIPaper.Author,
            p.AIPaper.Abstract)]]
  ∪for r in src2.Researcher, t in src2.Texts
  where r.Article.T=t.T
  return[Paper=[Title=r.Article.Title,
       Author=SKLM129(r.Article.Title, r.Article.Abstract, t.Txt),
       Abstract=r.Article.Abstract,
       K=SKLM130 (r.Article.Title, r.Article.Abstract, t.Txt)]]}
Rule24: SKLM med.1 ()= λ ().{
 for p in src1.Papers
  return[Text=[K=SKLM125(p.DBPaper.Title, p.DBPaper.Author,
       p.DBPaper.Abstract),
       Txt=SKLM127 (p.DBPaper.Title, p.DBPaper.Author,
       p.DBPaper.Abstract)]]
 ∪for p in src1.Papers
  return[Text=[K=SKLM126(p.AIPaper.Title, p.AIPaper.Author,
       p.AIPaper.Abstract),
       Txt=SKLM128(p.AIPaper.Title, p.AIPaper.Author,
       p.AIPaper.Abstract)]]
 ∪for r in src2.Researcher, t in src2.Texts
   where r.Article.T=t.T
   return[Text=[K=SKLM130(r.Article.Title, r.Article.Abstract, t.Txt),
       Txt=t.Txt]]}
```

图 6 映射规则集合

第 3 步 推导生成重写后的查询方案

输入:全局查询 Q,约束集合 C,映射规则集合 M_{rs}。

输出:重写后的查询重写方案 Q_{src}。

依据文献[8],先把全局查询集合中的各个查询 Q_i 转换成数据源查询集合;再重写约束集合中的约束,以附加的方式解决包含 Skolem 函数的等价关系,构造新的查询表达式并添加到 Q_{src} 中去,同时删除 Q_{src} 中不满足条件的源查询,然后产生能够产生相同结果的最小查询,如果原全局查询是嵌套的,则把它们重写组合成一个嵌套查询;最后对 Q_{src} 进行查询优化和聚合,生成重写后的查询重写方案 Q_{src}。例 1 产生的查询重写方案如图 7 所示。

3.2 算法分析

本文研究的目标是对原有算法的功能进行扩展,使其能够适应复杂模式匹配,而不是在性能上对原有算法进行改进,但也不应该为扩展的部分付出过大的代价。本文所扩展的部分是在算法的第 1 步和第 2 步,设每个数据源中元素的个数平均为 m 个,共有 n 个数据源,每个中介模式中原子元素在同一个数据源中的映射平均有 r 个,那么第 1 步建立数据源数据结构可以在 $O(m \times n)$ 时间内完成。第 2 步主要集中在数据源元素相互组合的匹配上,由于我们在第 1 步中已经通过 PJoin_ID, NJoin_ID 为每个元素建立起了联系,在这一步中只需要通过检查每个元素的数据结构信息,就可以完成相互的组合,这部分时间同原算法相同。设每个中介模式中原子元素在同一个数据源中的映射平均有 r 个,那我们要为 exists 子句的每个表达式建立 r 个映射规则,假设原算法在第 2 步中的时间复杂度为 $O(x)$,那么改进后的时间复杂度就为 $r \times O(x)$,根据文献[10]的经验,一般取 $r = 1.2$,那我们在第 2 步中时间复杂是原算法的 1.2 倍。第 3 步方法同原算法相同,时间复杂度也相同。对比发现本文对于文献[8]中算法所增加的时间复杂度仅为 $O(m \times n) + 0.2 \times O(x)$,这是一个多项式时间,并且这部分时间占整个查询重写过程是很小的一部分,因此改进的算法在性能上并没有实质性的额外开销。

```
ReQ1:
 for p in src1.Papers
 return [Title=p.DBPaper.Title, Author=p.DBPaper.Author,
         Abstract=p.DBPaper.Abstract, Txt=null]
ReQ2:
 for p in src1.Papers
 return [Title=p.AIPaper.Title, Author=p.AIPaper.Author,
         Abstract=p.AIPaper.Abstract, Txt=null]
ReQ3:
 for r in src2.Researcher, t in src2.Texts
 where r.Article.T=t.T
 return [Title=r.Article.Title, Author=null,
         Abstract=r.Article.Abstract, Txt=t.Txt]
```

图 7　查询重写方案

4 结语

本文提出了一种改进的 XML 查询重写算法,相对于原算法,改进的算法不但利用了其重写的思想,而且增加了对查询中引入模式匹配问题的解决方案,扩大了原查询重写算法的应用范围;但是解决模式匹配问题采用的方式表达能力是有限的,不能解决更加复杂的匹配问题。在今后的研究中将集中于如何能够在现在的基础上进一步寻找合理方案,来支持更加复杂的模式匹配,扩大算法的使用范围和减少额外开销。

参考文献:

[1] HALEVY A, RAJARAMAN A, ORDILLE J. Data integration: The teenage years[C]// VLDB. New York: ACM, 2006: 9 - 16.

[2] LENZERINI M. Data integration: A theoretical perspective [C]// The 21st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems. New York: ACM, 2002: 233 - 246.

[3] HALEVY A Y. Answering queries using views: A survey[J]. Very Large Database Journal, 2001, 10(4): 270 - 294.

[4] LEVY I, MILLSTEIN T. Navigational plans for data integration [C]// Proceedings of the Sixteenth National Conference on Artificial Intelligence and the Eleventh Innovative Applications of Artificial Intelligence Conference Innovative Applications of Artificial Intelligence. Menlo Park, CA, USA: AAAI, 1999: 67 - 73.

[5] GARCIA-MOLINA H, PAPAKONSTANINOU Y, QUASS D, *et al.* The TSIMMIS approach to mediation: Data model and languages[J]. Journal of Intelligent Information Systems, 1997, 8(2): 117 - 132.

[6] KIRK T, LEVY A Y, SAGIV Y, *et al.* The information manifold [C]// Proceedings of the AAAI 1995 Spring Symposium on Information Gathering from Heterogeneous, Distributed Environments. Menlo Park, CA, USA: AAAI, 1995: 85 - 91.

[7] Extensible Markup Language (XML) 1.0[EB/OL]. [2008 - 09 - 01]. http://www.w3.org/TR/xml.

[8] YU C, POPA L. Constraint-based XML query rewriting for data integration[C]// International Conference on Management of Data. New York: ACM, 2004: 371 - 382.

[9] 金鑫,金远平.一种改进的基于约束关系的 XML 查询重写算法研究[J].计算机研究与发展,2007,44(5):845 - 852.

[10] CLUET S, VELTRI P, VODISLAV D. Views in a large-scale XML repository[J]. The VLDB Journal, 2002, 11(3): 238 - 255.

(上接第 848 页)

4 结语

本文使用 χ^2 统计对用户评论中的特征项个数进行了压缩,在尽量减少信息缺失的前提下实现了特征降维,结合用户评论内容所具有的个性特征,提取出其中的产品特征、情感词等内容作为分类特征,并使用支持向量机对用户评论进行分类,获取对用户有用的优秀评论。试验结果表明该方法的分类效果令人满意。

参考文献:

[1] KU L-W, CHEN H-H. Mining opinions from the Web: Beyond relevance retrieval[J]. Journal of the American Society for Information Science and Technology, 2007, 58(12): 1838 - 1850.

[2] POPESCU A-M, ETZIONI O. Extracting product features and opinions from reviews[C]// Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP 2005). Morristown, NJ, USA: Association for Computational Linguistics, 2005: 339 - 346.

[3] HU MINQING, LIU BING. Mining and summarizing customer reviews[C]// Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York: ACM, 2004: 168 - 177.

[4] BOUNIE D, BOURREAU M, GENSOLLEN M, *et al.* The Effect of Online Customer Reviews on Purchasing Decisions: the Case of Video Games[EB/OL]. [2008 - 09 - 01]. http://e.darmon.free.fr/workcommed/papers/ bounie_bourreau_ gensollen_ waldbroeck_2_nice.pdf.

[5] KU L-W, LIANG Y-T, CHEN H-H. Opinion extraction, summarization and tracking in news and blog corpora[C]// Proceedings of AAAI-2006 Spring Symposium on Computational Approaches to Analyzing Weblogs. [S.l.]: AAAI, 2006: 100 - 107.

[6] AGICHTEIN E, GRAVANO L. Snowball: Extracting relations from large plain-text collections[C]// ACM International Conference on Digital Libraries. New York: ACM, 2000: 85 - 94.

[7] 熊忠阳,张鹏招,张玉芳. 基于 χ^2 统计的文本分类特征选择方法的研究[J]. 计算机应用,2008,28(2):513 - 518.

[8] ABNEY S. Bootstrapping[C]// Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-02). Morristown, NJ, USA: Association for Computational Linguistics, 2002: 360 - 367.