ESSK:一种计算点击流相似度的新方法

刘 嘉 祁 奇 陈振宇 惠成峰
(计算机软件新技术国家重点实验室 南京 210093) (南京大学软件学院 南京 210093)

摘 要 用户点击流信息被广泛应用于Web使用信息挖掘中。点击流相似度常用于用户会话分类和聚类。SSK(String Subsequence Kernel)最初被用于计算字符串相似度,后被引入计算点击流相似度,并成为目前常用方法之一。SSK选择两个字符串所有长度为k的子序列生成特征空间。单一k的选择往往存在特征数不足的问题,从而难以获得足够精确的点击流相似度。因此,提出一种新的点击流相似度计算方法ESSK(Extended String Subsequence Kernel)。ESSK采用所有子序列生成特征空间以解决SSK存在的问题。同时提出一种高效计算ESSK的算法,以降低计算复杂度。实验表明,ESSK比SSK更精确,比其它方法具有更高的区分度,因此更适合点击流相似度分析和应用。

关键词 点击流相似度,算法设计,计算复杂度
中图法分类号 TP301 **文献标识码** A

ESSK:A New Approach to Compute Clickstream Similarity

LIU Jia QI Qi CHEN Zhen-yu HUI Cheng-feng
(State Key Laboratory for Novel Software Technology,Nanjing 210093,China)
(Software Institute,Nanjing University,Nanjing 210093,China)

Abstract Clickstream is widely used in Web usage mining. Clickstream similarity is usually used to classify or cluster Web user sessions. SSK(string subsequence kernel) is an approach for computing string similarity originally. Then it is introduced to compute clickstream similarity and becomes one of the most popular methods. It selects all subsequences of length k of two strings to generate the feature space. A single value of k may cause a problem that the number of features is not enough to get an accurate clickstream similarity. So,a new approach to compute clickstream similarity ESSK (extended string subsequence kernel) was proposed. ESSK generates the feature space by all subsequences to solve the problem of SSK. To reduce the complexity of computation,an effective algorithm to compute ESSK was proposed. An experiment indicates that ESSK is more accurate than SSK and has a higher discrimination than other approaches. So it is more suitable to compute clickstream similarity.

Keywords Clickstream similarity,Design of algorithm,Computation complexity

1 引言

随着互联网的快速发展,Web使用信息挖掘[1,2]得到广泛应用。Web使用信息挖掘可以为用户提供个性化的内容,从而提高用户满意度。点击流通常采用用户浏览的元素(如页面)序列。点击流分析可以对Web用户行为进行直观展现,因而被广泛地用于Web使用信息挖掘[3,4]。

计算点击流相似度是分析点击流数据的一种基本方法,可以进一步用于用户分类和聚类[5]。计算点击流相似度可以发现用户的相似行为,从而找到一个相似用户群。有效的相似用户群发现依赖于精确的点击流信息和相似度度量方法。本文关注后者的改进工作。如果点击流相似度的区分度不高,就难以区分相似和不相似的点击流信息,从而难以发现有效的相似用户群。

SSK[6]是一种常用的点击流相似度计算方法。SSK最初用于计算文本的相似度,后被用于计算点击流相似度[7]。SSK选择两个字符串所有长度为k的子序列来生成特征空间,然后将这两个字符串映射为特征空间中的两个向量。单一的k选择往往使得SSK的特征数不够,从而导致计算得到的相似度不够精确。使用SSK计算点击流相似度,k的选择往往对结果有很大影响。当k取值为1时,点击流相似度偏高;当k取值大于1时,点击流相似度偏低。本文提出一种新的点击流相似度计算方法ESSK。ESSK是SSK的一个扩展,用所有的子序列生成向量空间。ESSK不需要设置长度k的值,因特征维度更高,故计算得到的相似度更准确。本文的贡献如下:

到稿日期:2011-07-22 返修日期:2011-11-09 本文受国家自然科学基金(10871091),教育部人文社科基金(10YJC870020,10YJC630283)资助。

刘 嘉(1976-),男,博士生,讲师,主要研究方向为电子商务、推荐系统、社会化网络;祁 奇(1987-),男,硕士生,主要研究方向为Web挖掘和个性化推荐;陈振宇(1978-),男,博士,副教授,主要研究方向为软件行为分析、测试用例演化、推荐系统;惠成峰(1988-),男,硕士生,主要研究方向为个性化推荐系统。

1)提出一种新的点击流相似度度量方法 ESSK。

2)提出一种高效算法完成 ESSK 计算。

3)实验验证了 ESSK 的有效性。

本文第 2 节介绍相关工作,包括"编辑距离"、"最长相同子串"等;第 3 节描述 SSK 和 ESSK,并提出一种高效计算 ESSK 的算法;第 4 节通过实验验证了 ESSK 比 SSK 及其它方法更适合计算点击流相似度;最后总结本文。

2 相关工作

研究者们提出很多方法来度量点击流相似度,"编辑距离"是其中最流行的方法之一。"编辑距离"以一个字符串改变为另一字符串需要的最少操作次数来度量两个字符串的距离。通常,允许的操作包括增加、删除和替换。"最长相同子串"是一种"编辑距离"的变异方法,可以和"编辑距离"采取相同的计算方法。"最长相同子串"计算两个字符串的相同子串的最大长度。

例如,字符串"abc"和 "cba"的"编辑距离"是 2,字符串"abc"和"dbe"的"编辑距离"也是 2。字符串"abc"和"cba"的"最长相同子串"是 1,字符串"abc"和"dbe"的"最长相同子串"也是 1。如果"a","b","c""d","e"代表 5 个 Web 页面,那么"abc","cba","dbe"代表 3 条点击流数据。我们认为"abc"和"cba"比"abc"和"dbe"更相似,因为"abc"和"cba"代表两个用户浏览了 3 个相同的页面,而"abc"和"dbe"代表两个用户仅仅浏览了一个相同的页面。从上述例子可以发现,"编辑距离"和"最长相同子串"的区分度往往不高。

研究者们在"编辑距离"和"最长相同子串"的基础上提出了多种扩展。例如,R. Kothari 等提出了根据页面的共同出现频率来计算"编辑距离"中的"替换"操作的代价[8]。A. Banerjee 等提出了根据页面停留时间计算权重的带权重的"最长相同子串"方法[9]。还有一些算法根据序列对准的方法计算点击流的相似度[10-12]。序列对准的方法可以根据时间信息、页面 URL 信息调整"编辑距离"的操作代价。但是这些方法并不能够从本质上克服"编辑距离"和"最长相同子串"在计算点击流相似度时存在的区分度不足的缺点。本文在 SSK 的基础上进行扩展提出的 ESSK 方法与"编辑距离"和"最长相同子串"的思想不同,通过抽取点击流的特征向量的方法从本质上提高了区分度。

3 计算点击流相似度的方法

3.1 String Subsequence Kernel

SSK[6]是一种用余弦相似度来计算两个字符串相似度的方法。它选择两个字符串的所有长度为 k 的子序列来生成向量空间,然后把这两个字符串映射为向量空间里的两个向量。两个向量的相似度可以直接用余弦相似度计算。一个字符串的长度为 k 的子序列是由任意 k 个字符串中的字符(不要求连续)以原来的顺序排列成的序列。当把字符串映射到向量空间中时,考虑连续的序列权重更高,所以每个子序列的权重是衰减因子 λ 的 n 次方(n 代表子序列在字符串中的长度)。衰减因子 λ 满足 $0<\lambda<1$。

例 1 考虑两个字符串"cat"和"car"。若 k 值设为 2,共获得 5 个特征,分别为"ca","ct","at","cr"和"ar",然后把"cat"和"car"映射为向量空间里的两个向量,如表 1 所列。

表 1 SSK 特征向量

	ca	ct	at	cr	ar
φ(cat)	λ^2	λ^3	λ^2	0	0
φ(car)	λ^2	0	0	λ^3	λ^2

字符串"cat"和"car"的相似度 S_{SSK}(cat,car)的计算公式为

$$S_{SSK}(\text{cat},\text{car})=\frac{K_{SSK}^{k=2}(\text{cat},\text{car})}{\sqrt{K_{SSK}^{k=2}(\text{cat},\text{cat})}\sqrt{K_{SSK}^{k=2}(\text{car},\text{car})}}=\frac{(\varphi(\text{cat}),\varphi(\text{cat}))}{\sqrt{(\varphi(\text{cat}),\varphi(\text{cat}))}\sqrt{(\varphi(\text{car}),\varphi(\text{car}))}}=\frac{\lambda^4}{2\lambda^4+\lambda^6}=\frac{1}{2+\lambda^2} \tag{1}$$

SSK 选取所有长度为 k 的子序列生成特征空间,而丢弃了其他长度的子序列,计算得到的相似度往往由于特征数不足而不够精确。下面一个极端情况下的例子证明了我们的看法。

例 2 考虑两个字符串"ab"和"ba"。若 k 值设为 1,特征为"a"和"b",S_{SSK}(cat,car)=1。若 k 值设为 2,特征为"ab"和"ba",S_{SSK}(cat,car)=0。当 $k=1$ 时,丢弃了字符串"ab"和"ba"间的不同特征;当 $k=2$ 时,丢弃了字符串"ab"和"ba"间的相同特征。

采用 SSK 计算点击流相似度时,若 k 值为 1,则丢弃了顺序信息,从而往往丢弃了两条点击流数据的一些不同特征。若 k 值大于 1,则丢弃了长度小于 k 的相同子序列,从而丢弃了两条点击流数据的一些相同特征。综上所述,采用 SSK 计算点击流相似度时,由于特征数不够,可能导致一些不同特征或相同特征被丢弃的问题。

3.2 Extended String Subsequence Kernel

本文提出的方法 ESSK 选择所有子序列生成向量空间,解决了 SSK 存在的特征数不够的问题。ESSK 子序列的权重计算方法 SSK 相同。表 2 展示了字符串"ab"和"ba"的特征向量。S_{ESSK}(ab,ba)的计算见式(2)。

表 2 ESSK 特征向量

	a	b	ab	ba
φ(ab)	λ	λ	λ^2	0
φ(ba)	λ	λ	0	λ^2

$$S_{ESSK}(\text{ab},\text{ba})=\frac{K_{ESSK}(\text{ab},\text{ba})}{\sqrt{K_{ESSK}(\text{ab},\text{ab})}\sqrt{K_{ESSK}(\text{ba},\text{ba})}}=\frac{(\varphi(\text{ab}),\varphi(\text{ba}))}{\sqrt{(\varphi(\text{ab}),\varphi(\text{ab}))}\sqrt{(\varphi(\text{ba}),\varphi(\text{ba}))}}=\frac{2\lambda^2}{2\lambda^2+\lambda^4}=\frac{2}{2+\lambda^2} \tag{2}$$

采用 ESSK 计算两个长字符串的相似度时,由于特征比较多,因此直接计算的复杂度比较高。可以用对 $K_{SSK}(s,t)$求和的方法计算 $K_{ESSK}(s,t)$,见式(3)。

$$K_{ESSK}(s,t)=\sum_{d=1}^{\min(|s|,|t|)}K_{SSK}^{k=d}(s,t) \tag{3}$$

在文献[6]中,优化后的 $K_{SSK}^{k=d}(s,t)$的计算复杂度为 $O(d|s||t|)$。若采用式(3),$K_{SSK}(s,t)$的计算复杂度近似为 $\min(|s||t|^3,|s|^3|t|)$。下面,我们提出了一种高效计算 ESSK 的算法,计算 $K_{ESSK}(s,t)$的复杂度可以降低到 $O(|s||t|)$。

先介绍一些引自文献[6]中的定义。Σ 是一个有限的字

符集。一个字符串是由有限的属于 Σ 的字符(可重复)组成的序列,包括空串。对于字符串 s 和 t,$|s|$ 被定义为字符串 $s=s_1\cdots s_{|s|}$ 的长度。st 代表由字符串 s 和 t 连接形成的新字符串。字符串 $s[i:j]$ 代表字符串 s 的子串 $s_i\cdots s_j$。如果 u 满足条件:存在 $i=(i_1,\cdots,i_{|u|})(1\leqslant i_1<\cdots<i_{|u|}\leqslant|s|)$ 满足 $u_j=s_{ij}$ $(j=1,\cdots,|u|)$,即 $u=s_{[i]}$,则 u 是 s 的一个子序列。子序列 u 在 s 中的长度 $l(i)=i_{|u|}-i_1+1$。

定义 Σ_n 为所有长度为 n 的字符串的集合,Σ^* 为所有字符串的集合:

$$\Sigma^*=\bigcup_{n=0}^{\infty}\Sigma^n \tag{4}$$

定义特征空间 $F^*=R^{\Sigma^*}$。对于字符串 s,特征映射 φ 定义了每个属于 Σ^* 的 u 在 s 中的权重 $\varphi_u(s)$。

$$\varphi_u(s)=\sum_{i:u=s_{[i]}}\lambda^{l(i)} \tag{5}$$

定义:

$$\begin{aligned}K_{ESSK}(s,t)&=\sum_{u\in\Sigma^*}(\varphi_u(s),\varphi_u(t))\\&=\sum_{u\in\Sigma^*}\sum_{i:u=s_{[i]}}\lambda^{l(i)}\sum_{j:u=t_{[j]}}\lambda^{l(j)}\\&=\sum_{u\in\Sigma^*}\sum_{i:u=s_{[i]}}\sum_{j:u=t_{[j]}}\lambda^{l(i)+l(j)}\end{aligned} \tag{6}$$

$$K'(s,t)=\sum_{u\in\Sigma^*}\sum_{\substack{i:u=s_{[i]}\\ i|u|=s_{|s|}}}\lambda^{l(i)}\sum_{\substack{j:u=t_{[j]}\\ j|u|=t_{|t|}}}\lambda^{l(j)} \tag{7}$$

$$K''(s,t)=\sum_{u\in\Sigma^*}\sum_{i:u=s_{[i]}}\sum_{j:u=t_{[j]}}\lambda^{|s|+|t|-i_1-j_1+2} \tag{8}$$

式(9)—式(11)是 K,K' 和 K'' 之间的关系。在式(10)和式(11)中,x 和 y 代表两个字符。

$$K_{ESSK}(s,t)=\sum_{i=1}^{|s|}\sum_{j=1}^{|t|}K'(s[1,i],t[1,j]) \tag{9}$$

$$K'(sx,ty)=\begin{cases}0, & x\neq y\\ \lambda^2[1+K''(s,t)], & x=y\end{cases} \tag{10}$$

$$K''(sx,ty)=K'(sx,ty)+\lambda K''(sx,t)+\lambda K''(s,ty)-\lambda^2K''(s,t) \tag{11}$$

式(9)对 K 进行分解。式(10)中,若 x,y 不等,则不存在 u 满足最后一个字符既为 x 又为 y。若 x,y 相等,那么满足“既为 sx 的一个子序列,也为 ty 的一个子序列,同时最后一个字符为 x 也为 y”的 $u(u=x$ 除外$)$与满足“既为 s 的一个子序列,也为 t 的一个子序列”的 u' 一一对应。式(11)满足“既为 sx 的一个子序列,也为 ty 的一个子序列”的所有 u 的集合 U 可以写成 $U=U_1+U_2+U_3-U_4$。其中 U_1 是“满足既为 sx 的一个子序列,也为 ty 的一个子序列,同时最后一个字符为 x 也为 y”的 u 的集合。U_2 是满足“既为 sx 的一个子序列,也为 t 的一个子序列”的 u 的集合。U_3 是满足“既为 s 的一个子序列,也为 ty 的一个子序列”的 u 的集合。U_4 是满足“既为 s 的一个子序列,也为 t 的一个子序列”的集合。

编程算法如算法 1 所示,计算 $K_{ESSK}(s,t)$ 的复杂度为 $O(|s||t|)$。

算法 1 ESSK 算法

对于两个字符串 $V=v_1v_2\cdots v_m$ 和 $W=w_1w_2\cdots w_n$,K(V,W)计算伪代码如下:

```
创建矩阵 D^(m+1,n+1)
For i=1 to m+1
    For j=1 to n+1
        D(i,j) have two values:
          D(i,j).value1=K'(V[1,i]),W[1,j])
          D(i,j).value2=K''(V[1,i]),W[1,j])
For i=1 to m+1
        初始化 D(i,1).value1=0,D(i,1).value2=0
For j=1 to n+1
        初始化 D(1,j).value1=0,D(1,j).value2=0
初始化 sum=K_ESSK(V,W)=0
For i=2 to m+1
    For j=2 to n+1
        D(i,j).value1= { 0,                      v_i≠w_j
                       { λ²[1+D(i-1,j-1)],       v_i=w_j
        sum=sum+D(i,j)
        D(i,j).value2=D(i,j).value1+λD(i-1,j).value2+λD(i,j-1).value20-λ²D(i-1,j-1).value2
```

4 实验

4.1 实验目标

本实验的目标是验证以下两个问题:

1)ESSK 是否比“编辑距离”和“最长相同子串”具有更高的区分度?

2)ESSK 是否比 SSK 更精确?具体表现为 ESSK 计算得到的相似度值比 SSK($k=1$)计算得到的相似度值低,同时比 SSK($k>1$)计算得到的相似度值高。

4.2 实验设计

实验所用数据来自 NASA Kennedy Space Center Server[13]。我们随机选取了一天的服务器日志并进行初始化后得到 402 条点击流数据。为了计算区分度,随机选取一条点击流数据,然后计算剩余的 401 条点击流数据与它的相似度,得到 401 个相似度值。

4.3 实验结果

分别采用“编辑距离”、“最长相同子串”、SSK 和 ESSK 来计算点击流的相似度。SSK 和 ESSK 中的参数 λ 被设置为 0.5(在实际应用中,λ 的值需要经过训练得到。在本实验中,λ 的值只影响相似度的绝对值,而不影响相对大小,因此不影响区分度)。由“编辑距离”和“最长相同子串”计算得到的相似度被标准化为 0 到 1 之间,见式(12)和式(13)。

$$S_{\text{编辑距离}}=1-\frac{\text{edit}(s,t)}{\max(s,t)} \tag{12}$$

$$S_{\text{最长相同子串}}=\frac{2LCS(s,t)}{|s|+|t|} \tag{13}$$

图 1—图 3 分别是由“编辑距离”、“最长相同子串”和 ESSK 计算得到的相似度的频率分布图。横轴代表相似度,纵轴代表频数。因为有 101 条点击流数据和选取的参照点击流数据没有共同页面,所以计算得到的相似度结果至少包含 101 个 0。

若两条点击流数据和选取的参照点击流数据的相似度相同,则认为不能区分,记为“0”;相反,则认为可以区分,记为“1”。一共有 401 个相似度的值,两两比较共得到 80200 个“1”或“0”。我们定义区分度为“1”所占的百分比。计算得到“编辑距离”、“最长相同子串”和 ESSK 的区分度分别为 0.565,0.894 和 0.920。实验结果表明,ESSK 比“编辑距离”和“最长相同子串”具有更高的区分度。

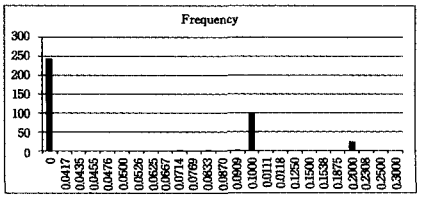

图 1 编辑距离

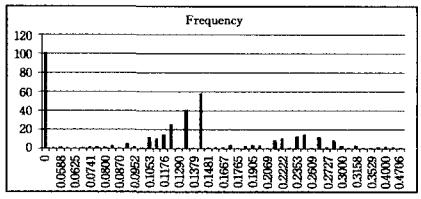

图 2　最长相同子串

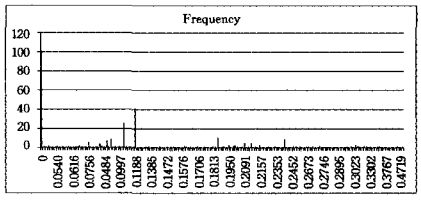

图 3　ESSK

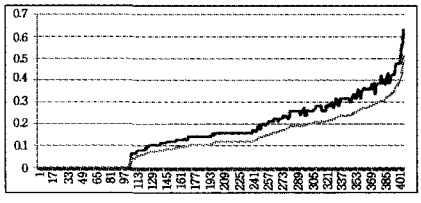

图 4　SSK(k=1)&ESSK

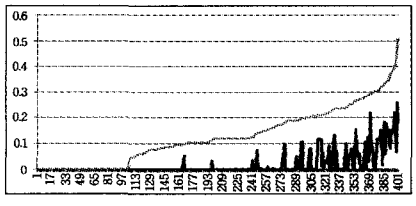

图 5　SSK(k=2)&ESSK

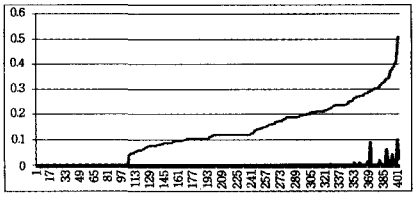

图 6　SSK(k=3)&ESSK

图 4—图 6 比较了由 SSK 和 ESSK 计算得到的相似度的值。横轴是点击流的 id,纵轴是相似度。浅色线代表 ESSK,深色线代表 SSK。点击流的 id 根据相似度的值从小到大排列,以便观察实验结果。从图 4 可以看出,当 k=1 时,SSK 计算得到的相似度值比 ESSK 大。从图 4 和图 5 可以看出,当 k=2 和 k=3 时,SSK 计算得到的相似度值比 ESSK 小。实验结果表明,ESSK 的相似度值稳定适中,更适合计算点击流相似度。

结束语　本文对 SSK 进行扩展,提出一种新的计算点击流相似度的方法 ESSK,并提出一种高效计算 ESSK 的算法。本文通过理论分析和实验结果表明,ESSK 比传统方法("编辑距离"、"最长相同子串"和 SSK)更适合计算点击流相似度。

参 考 文 献

[1] Srivastava J, Cooley R, Deshpande M, et al. Web usage mining: Discovery and applications of usage patterns form Web data[C]// SIGKDD Explorations. 2000

[2] 郭岩,白硕,于满泉. Web 使用信息挖掘综述[J]. 计算机科学,2005,32(1):1-7

[3] Cooley R W. Web Usage Mining: Discovery and Application of Interesting Patterns from Web Data[D]. Minnesota, USA: University of Minnesota, 2000

[4] 张波,巫莉莉,周敏. 基于 Web 使用挖掘的用户行为分析[J]. 计算机科学,2006,33(8):213-214

[5] 马超,沈微. 基于闭合有间隔频繁子序列的点击流聚类[J]. 计算机工程,2010(23):72-75

[6] Lodhi H, Saunders C, Shawe-Taylor J, et al. Text Classification Using String Kernels[J]. The Journal of Machine Learning Research, 2002(2):419-444

[7] Koch D. Study of the discrepancy between client-and server side logging of clickstreams[M]. Stockholm: Royal Technical Institute, 2006

[8] Kothari R, Mittal P, Jain V, et al. On Using Page Cooccurrences for Computing Clickstream Similarity[C]// Proceedings of SIAM International Conference on Data Mining. 2003

[9] Banerjee A, Ghosh J. Clickstream clustering using weighted longest common subsequences[C]// Proceedings of the Web Mining Workshop at the SIAM Conference on Data Mining. 2001

[10] Hay B, Wets G, Vanhoof K. Clustering navigation patterns on a website using a sequence alignment method[C]// Proceedings 17th Conference on Artificial Intelligence, Intelligent Techniques for Web Personalization. 2001

[11] Wang W, Zaiane O R. Clustering Web sessions by sequence alignment[C]// Proceedings of DEXA. 2002

[12] Gunduz S, Ozsu M T. A Web Page Prediction Model Based on ClickStream Tree Representation of User Behavior[C]// Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2003

[13] NASA Kennedy Space Center Log[EB/OL]. http://ita.ee.lbl.gov/html/contrib/NASA-HTTP.html

(上接第 119 页)

[7] Brun C, Musset J, Toulme A. EMF Compare[EB/OL]. http://wiki.eclipse.org/index.php/EMF_Compare, 2007

[8] Altmanninger K. Models in conflict-Towards a semantically enhanced version control system for models[M]. Nashville. TN, United states, 2008

[9] Brunet G, Chechik M, Easterbrook S, et al. A manifesto for model merging[C]//GaMMa '06. New York, NY, USA: ACM, 2006

[10] 尹剑飞,郭荷清,彭新一. 基于模型转换实现行为协议的研究[J]. 计算机工程,2005,31(1):31-32

[11] Yin Jian-fei, H G X P. Pattern Semantic Link: A Reusable Pattern Representation in MDA Context[C]// ICDCIT 2004. LNCS 3347, 2004:310-317

[12] Peltier J, B G G M. MTRANS: A General Framework Based on XSLT for Model Transformations[C]// Proceedings of the Workshop on Transformations in UML. 2001

[13] Gray J, Z Y L J. Model-Driven Program Transformation of a Large Avionics Framework[C]// GPCE 2004. LNCS 3286, 2004:361-378

[14] 任胜兵. 基于图变换的可视化层次用例建模及演化方法研究[D]. 长沙:中南大学,2007

[15] 鲍爱华. 语义 Web 环境下组合服务演化方法及其关键技术研究[D]. 长沙:国防科学技术大学,2009