Dijkstra算法中的多邻接点与多条最短路径问题

王树西 李安渝

(对外经济贸易大学信息学院 北京 100029) (对外经济贸易大学电子商务研究所 北京 100029)

摘 要 Dijkstra算法是图论中求取最短路径的经典算法。列举并分析了Dijkstra算法及其伪码,为了深刻理解Dijkstra算法,列举了几种错误观点并加以纠正。分析发现,根据Dijkstra算法,最短路径上的某个顶点的前面,可能有多个邻接点;从开始点到某个顶点之间,可能存在多条权重相同的最短路径。对于上述多邻接点问题与多条最短路径问题,Dijkstra算法并没有涉及。分析了多邻接点问题与多条最短路径问题的成因,提出解决方案,对Dijkstra算法进行了改进,给出了改进之后的算法与伪码,分析了算法的时间复杂度,并用c语言编码实现。实验结果表明,改进之后的Dijkstra算法可以有效解决多邻接点问题与多条最短路径问题。

关键词 Dijkstra算法,多邻接点,多条最短路径,时间复杂度

中图法分类号 TP392 **文献标识码** A

Multi-adjacent-vertexes and Multi-shortest-paths Problem of Dijkstra Algorithm

WANG Shu-xi LI An-yu

(Information Academy University of International Business and Economics, Beijing 100029, China)

(Institute of Electronic Commerce, University of International Business and Economics, Beijing 100029, China)

Abstract Dijkstra Algorithm is one of the most classical algorithms to solve the shortest path problem. This paper listed and analyzed Dijkstra Algorithm and its pseudo-code. To deeply understand Dijkstra Algorithm, listed several error views and rectified them. Through analyzing Dijkstra Algorithm, there are maybe multiple pre-adjacent vertexes for a vertex in one shortest path, and there are maybe multiple shortest paths with the same weight. Regrettably, Dijkstra Algorithm does not solve the above problems. To solve the above problems and improve Dijkstra Algorithm, we analyzed its causes, proposed an algorithm, gave its pseudo-code and programmed with c language, and analyzed the time complexity of this algorithm. Experimental results show that the improved Dijkstra algorithm can effectively solve the problem of multi-adjacent-vertexes and multi-shortest paths.

Keywords Dijkstra algorithm, Multiple pre-adjacent vertexes, Multiple shortest paths, Time complexity

1 引言

最短路径问题,是图论研究中的一个经典问题,它旨在寻找图中两结点之间的最短路径,也就是沿此路径上各边的权值总和(路径长度)达到最小。

Dijkstra[1]给出的算法,是图论中求取最短路径的经典算法。Dijkstra算法应用的范围非常广泛,例如:在多点路由上的应用、在测绘科学中的应用、在物流运输最短路径中的应用、在智能交通系统中的应用、在高速公路联网收费中的应用等。

本文结构如下:综述了国内外对Dijkstra算法的研究现状;对Dijkstra算法进行了描述,给出了伪码并进行分析,提出用邻接矩阵表示图形更易于编程实现;通过表格模拟运行算法,并用c语言编程实现算法;列举了对Dijkstra算法的几种错误理解并进行纠正;重点分析了多邻接点问题与多条最短路径问题,包括成因,并给出解决方案;在上述工作的基础上,对Dijkstra算法进行了改进,给出了改进之后算法的伪码。对改进的算法进行实验,实验结果表明,改进之后的算法可以有效解决多邻接点问题以及多条最短路径问题。

2 国内外研究现状

近年来,国内对Dijkstra算法的研究非常活跃,重要期刊上发表了许多相关的论文。这些研究集中在两个方面:(1)对Dijkstra算法本身进行改进;(2)根据Dijkstra算法进行应用研究。国内研究偏重于Dijkstra算法的具体应用。

例如,王树西[2]研究了改进的Dijkstra最短路径算法及其应用;董俊[3]研究了改进Dijkstra算法在GIS导航应用中的最短路径搜索;刘国华[4]研究了基于Dijkstra距离的聚类算法及其在物流中的应用;郑四发[5]研究了复杂路网下多客户间最短路径的扇面Dijkstra算法;刘建美[6]研究了基于改

到稿日期:2013-08-13 返修日期:2013-11-22 本文受对外经济贸易大学信息学院基金,对外经贸大学:《信息系统建设与实施》案例研究(X12511),对外经贸大学:商务信息系统应用(X10017),对外经济贸易大学中央高校基本科研业务费专项资金(13YBLG02)资助。

王树西(1976—),男,博士,讲师,主要研究方向为电子商务,E-mail:wangshuxi2006@sina.com;李安渝(1963—),男,博士,教授,主要研究方向为电子商务。

进的 Dijkstra 算法的动态最短路计算方法；李敬贤[7]研究了求解震后最优路径的改进 Dijkstra 算法；周竞文[8]研究了基于 Dijkstra 距离剪枝的测地线求解算法；吴若伟[9]研究了基于 Dijkstra 算法的大型停车场最优泊车路径规划。

国外对 Dijkstra 算法的研究也很活跃，近年来相关论文有很多。国外的研究集中在 Dijkstra 算法本身的改进，以及 Dijkstra 算法的具体应用。应该说，国内外对 Dijkstra 算法的研究各有特色，但是都偏重算法优化及算法应用。

例如，Idwan[10]研究了用启发式算法对 Dijkstra 算法进行改进，并将其应用于大规模图形中求取最短路径；Medeiros[11]研究了 Dijkstra 算法在基于地形高程固定翼无人机运动规划中的应用；Gunkel[12]研究了 Dijkstra 最短路径算法在机器微裂纹检测中的应用；Li[13]研究了利用限制方向方法及二叉堆技术对 Dijkstra 算法进行优化，并在 WebGIS 中进行应用；Tintor[14]研究了利用分布式稀疏矩阵对 Dijkstra 算法进行优化，并在半透明光网络路由中进行应用；Kimoto[15]研究了 Dijkstra 三态互斥算法的时间复杂度；Bauer[16]研究了基于目标导向与多层次相结合的 Dijkstra 算法的加速技术。

3 Dijkstra 算法概述与分析

首先借用文献[17]的内容，对 Dijkstra 算法进行描述，给出伪码并进行分析，指出用邻接矩阵表示图形更加易于编程实现。通过表格模拟运行，用 c 语言编程实现，并对算法进行评论。

3.1 算法描述

Dijkstra 算法也称为双标号法。所谓双标号，也就是对图中的点 v_i 赋予两个标号$(P(v_i),\lambda_i)$：第一个标号 $P(v_i)$ 表示从起点 v_1 到 v_i 的最短路的长度，第二个标号 λ_i 表示在 v_1 到 v_i 的最短路上 v_i 前面一个邻点的下标，即用来表示路径，从而可对终点到始点进行反向追踪，找到 v_1 到 v_n 的最短路。Dijkstra 算法适用于每条边的权数都大于或等于零的情况。下面是 Dijkstra 算法的内容：

设 $G=\langle V,E,W\rangle$ 是 n 阶简单带权图，$w_{ij}\geqslant 0$。如果顶点 v_i 与 v_j 不相邻，令 $w_{ij}=\infty$，求 G 中顶点 v_1 到其余各顶点的最短路径(其实可以求任意两顶点之间的最短路径)，为此，先给出下面的定义：

(1)设 $l_i^{(r)*}$ 为顶点 v_1 到顶点 v_i 最短路径的权，若顶点 v_i 获得了标号 $l_i^{(r)*}$，称 v_i 在第 r 步获得了 p 标号 $l_i^{(r)*}$(永久性标号)，其中 $r\geqslant 0$。

(2)设 $l_j^{(r)}$ 为 v_1 到顶点 v_j 的最短路径权的上界，如果 v_j 获得 $l_j^{(r)}$，称 v_j 在第 r 步获得 t 标号 $l_j^{(r)}$(临时性标号)。

(3)设 $P_r=\{v|v$ 已获得 p 标号$\}$为第 r 步通过集。

(4)设 $T_r=V-P_r$ 为第 r 步未通过集，$r\geqslant 0$。

Dijkstra(标号法)的算法如下：

开始，$r\leftarrow 0$，v_1 获 p 标号：$l_1^{(0)*}=0$，$P_0=\{v_1\}$，$T_0=V-\{v_1\}$，$v_j(j\neq 1)$的 t 标号：$l_j^{(0)}=w_{ij}$。

①求下一个 p 标号顶点

设 $l_i^{(r)*}=\min\limits_{v_j\in T_{r-1}}\{l_j^{(r-1)}\}$，$r\geqslant 1$。将 $l_i^{(r)*}$ 标在相应顶点 v_i 处，表明 v_i 获得 p 标号，修改通过集和未通过集：$P_r=P_{r-1}\cup\{v_i\}$，$T_r=T_{r-1}-\{v_i\}$。

查 T_r：若 $T_r=\varnothing$，则算法结束，否则转②。

②修改 T_r 中各顶点的 t 标号

$l_j^{(r)}=\min\{l_j^{(r-1)},l_i^{(r)*}+w_{ij}\}$，$l_i^{(r)*}$ 是刚刚获得 p 标号顶点的 p 标号。令 $r\leftarrow r+1$，转①。

3.2 算法伪码

http://en.wikipedia.org/wiki/Dijkstra%27s_algorithm(维基百科)中，已经给出 Dijkstra 算法的伪码，现在列举如下。

```
1. function Dijkstra(Graph, source):
2.     for each vertex v in Graph:
3.         dist[v] := infinity;
4.         previous[v] := undefined;
5.     end for
6.     dist[source] := 0;
7.     Q := the set of all nodes in Graph;
8.     while Q is not empty:
9.  u := vertex in Q with smallest distance in dist[];
10.        remove u from Q;
11.        if dist[u]=infinity:
12.            break;
13.        end if
14.        for each neighbor v of u:
15.            alt := dist[u]+dist_between(u,v);
16.            if alt<dist[v]:
17.                dist[v] := alt;
18.                previous[v] := u;
19.                decrease-key v in Q;
20.            end if
21.        end for
22.    end while
23.    return dist;
24. endfunction
```

3.3 理解与分析算法、伪码

文献[2]提出："在 Dijkstra 标号法中，算法的退出机制是：查第 r 步未通过集 $T_r(r\geqslant 0)$：若 $T_r=\varnothing$，则算法结束。这种退出机制，对于无向带权联通图是有效的，但是在有向带权图中，如果两个点之间是不连通的，那么根据现有算法的退出机制，系统将无法停机而陷入死循环"。

应该指出，网上很多早期编写的 Dijkstra 算法程序确实没有考虑到这种情况。但是通过分析 Dijkstra 算法的伪码可以看出，这种情况已经得到了妥善处理。参考上述伪码的第 11－13 步骤，这 3 个步骤是处置这种情况的，从而避免系统陷入死循环。

文献[2]提出："Dijkstra 的标号法忽略了一个问题，那就是：多个顶点可以同时获得 p 标号"、"如果多个点的 t 标号相同，那么所有这些点同时获得 p 标号"。

这种说法是完全错误的。通过仔细分析 Dijkstra 算法可以看出，每次只能有一个顶点获得 p 标号(永久性标号)，即使多个顶点的 t 标号(临时性标号)相同，也只能逐个顶点获得 p 标号(永久性标号)。这并不影响算法最后的结果。

从 Dijkstra 算法描述以及伪码可以看出，Dijkstra 算法默认为最短路径上某个顶点只有一个前置的邻接点(参考上述伪码的 4、18、19 步骤)；而且上述伪码没有给出如何获得从源点出发到某一顶点的最短路径。

实际上，根据 Dijkstra 算法，最短路径上某个顶点可能有多个前置的邻接点，从源点出发到某一顶点之间，可能存在多条权重(长度)相同的最短路径。这是必须要解决的问题，也

是本文的重点内容。

对伪码进行分析可以看出，伪码并没有完全体现出 Dijkstra 算法中“永久性标号”思想。也就是说，最短路径上的顶点，应该通过“永久性标号”进行标注。

3.4 多条最短路径问题举例分析

下面给出一个例子，说明多个前置邻接点、多条最短路径的情况是客观存在的。

图 1 是一个有向带权图，下面用 Dijkstra 算法求取从 V_0 点到其余各点之间的最短路径及最短路径长度。通过表格模拟 Dijkstra 算法运行，如表 1 所列。

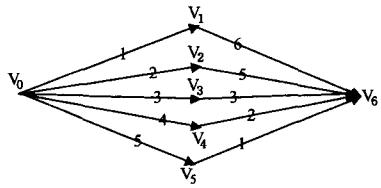

图 1 有向带权图 G1

表 1 Dijkstra 算法模拟运行表(求取多条最短路径)

	V_0	V_1	V_2	V_3	V_4	V_5	V_6
0	[0]	1/V_0	2/V_0	3/V_0	4/V_0	5/V_0	∞
1		[1]/V_0	2/V_0	3/V_0	4/V_0	5/V_0	7/V_1
2			[2]/V_0	3/V_0	4/V_0	5/V_0	7/V_1,V_2
3				[3]/V_0	4/V_0	5/V_0	6/V_3
4					[4]/V_0	5/V_0	6/V_3,V_4
5						[5]/V_0	6/V_3,V_4,V_5
6							[6]/V_3,V_4,V_5

通过表 1 的模拟运行，可以得到 V_0 点到其余各点之间的最短路径及最短路径长度。

V_0 点到 V_0 点的最短路径：V_0，长度为 0。

V_0 点到 V_1 点的最短路径：V_0V_1，长度为 1。

V_0 点到 V_2 点的最短路径：V_0V_2，长度为 2。

V_0 点到 V_3 点的最短路径：V_0V_3，长度为 3。

V_0 点到 V_4 点的最短路径：V_0V_4，长度为 4。

V_0 点到 V_5 点的最短路径：V_0V_5，长度为 5。

V_0 点到 V_6 点的最短路径有 3 条：

(1)$V_0V_3V_6$，长度为 6；

(2)$V_0V_4V_6$，长度为 6；

(3)$V_0V_5V_6$，长度为 6。

由此可以看出 V_6 点在最短路径上有 3 个前置的邻接点(V_3，V_4，V_5)。

从这个例子可以看出，多条最短路径、最短路径上某个顶点有多个前置的邻接点，这种情况是客观存在的，应该设法解决这个问题。

4 多条最短路径问题的求解

在分析 Dijkstra 算法及其源码的基础上，对 Dijkstra 算法进行改进，以解决多条最短路径问题。也就是说，根据 Dijkstra 算法，从源点开始到某个顶点之间如果有多条最短路径(路径长度相同)，那么求出所有的最短路径。

为了解决多条最短路径问题，应该从两点入手：改进算法；改进数据结构。

4.1 用邻接矩阵表示图

由于矩阵在计算机中易于储存和处理，因此可以利用矩阵将图表示在计算机中，而且还可以利用矩阵中的一些运算来刻画图的一些性质，研究图论中的一些问题。

为了更好地研究与编程实现 Dijkstra 算法，用邻接矩阵表示带权图(有可能是有向图或者无向图)。设简单图 $G=\langle V,E\rangle$，$V=\{V_1,V_2,\cdots,V_n\}$，$|E|=m$，称 n 阶方阵 $adjMatrix(G)=(a_{ij})$ 为带权图 G 的邻接矩阵(Adjacency Matrix)。其中，

$$a_{ij}=\begin{cases}W_{i,j}, & \text{weight of edge from } V_i \text{ to } V_j\\ \infty, & \text{there is no edge from } V_i \text{ to } V_j\\ 0, & \text{from } V_i \text{ to } V_i, i=j\end{cases}$$

邻接矩阵反映的是顶点与顶点之间的关系，矩阵的每个元素表示顶点之间边(有向边或者无向边)的权重。如果矩阵元素 a_{ij} 为∞($i\neq j$)，那么第 i 个顶点到第 j 个顶点之间没有边直接相连；如果矩阵元素 a_{ij} 为一个正整数 w($i\neq j$)，那么第 i 个顶点到第 j 个顶点之间有一条边(有向边或者无向边)，边的权重为 w；矩阵元素 a_{ii} 为 0。用邻接矩阵表示的图，任何一个顶点上都没有环，任何两点之间都没有平行边，是简单图。

矩阵中所有的元素，或者为 0，或者为∞，或者为正数，一般不存在负数。有向带权图对应的矩阵中，非 0、非∞的元素之和，表示有向带权图中所有有向边的个数；无向带权图对应的矩阵中，非 0、非∞的元素总的个数为偶数，元素个数之和的 1/2 为无向带权图中无向边的个数之和。

邻接矩阵的主对角线上的元素，全都是 0；无向带权图所对应的邻接矩阵关于主对角线对称。

之所以采用邻接矩阵表示带权图，是因为用邻接矩阵表示矩阵比较直观，并且 Dijkstra 算法需要多次修改顶点最短路径的长度，用邻接矩阵表示带权图，矩阵中的元素容易修改。用邻接矩阵表示带权图的缺点是，当带权图中的点比较多，带权图比较大的时候，邻接矩阵中的元素很多都是∞，也就是说矩阵可能过于稀疏。

下面分别用邻接矩阵表示有向带权图及无向带权图。

图 2 表示有向带权图 G1(见图 1)所对应的邻接矩阵。

	V_0	V_1	V_2	V_3	V_4	V_5	V_6
V_0	0	1	2	3	4	5	∞
V_1	∞	0	∞	∞	∞	∞	6
V_2	∞	∞	0	∞	∞	∞	5
V_3	∞	∞	∞	0	∞	∞	3
V_4	∞	∞	∞	∞	0	∞	2
V_5	∞	∞	∞	∞	∞	0	1
V_6	∞	∞	∞	∞	∞	∞	0

图 2 邻接矩阵 adj(G1)

图 3 表示煤气站 A，将给一居民区供应煤气，居民区各用户所在位置如图所示，铺设各用户点的煤气管道所需的费用(单位：万元)如图边上的数字所示。

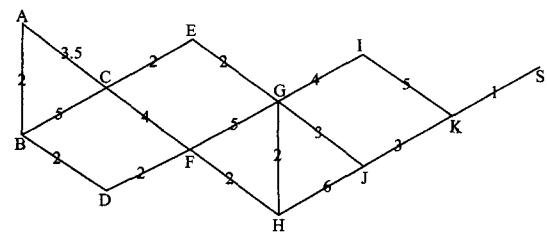

图 3 无向带权图 G2

图3是一个无向图，事实上，用有向图表示似乎更好，结果不受影响。下面用图4所示的邻接矩阵表示图3。

	A	B	C	D	E	F	G	H	I	J	K	S
A	0	2	3.5	∞	∞	∞	∞	∞	∞	∞	∞	∞
B	2	0	5	2	∞	∞	∞	∞	∞	∞	∞	∞
C	3.5	5	0	∞	2	4	∞	∞	∞	∞	∞	∞
D	∞	2	∞	0	∞	4	∞	∞	∞	∞	∞	∞
E	∞	∞	2	∞	0	∞	2	∞	∞	∞	∞	∞
F	∞	∞	4	4	∞	0	∞	2	∞	∞	∞	∞
G	∞	∞	∞	∞	2	∞	0	2	4	3	∞	∞
H	∞	∞	∞	∞	∞	2	2	0	∞	6	∞	∞
I	∞	∞	∞	∞	∞	∞	4	∞	0	∞	5	∞
J	∞	∞	∞	∞	∞	∞	3	6	∞	0	3	∞
K	∞	∞	∞	∞	∞	∞	∞	∞	5	3	0	1
S	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	1	0

图4　邻接矩阵 adj(G2)

上面的矩阵表示无向带权图 G2 所对应的邻接矩阵，这是一个较为稀疏的矩阵。

4.2　多条最短路径问题的深入分析

多条最短路径问题与多邻接点问题，实际上是同一个问题。深入分析多条最短路径问题与多邻接点问题，可以得到如下的一系列结论。

(1)每条最短路径可以表示为顶点连接而成的字符串。

(2)源点到源点之间最短路径的个数为1；源点到源点的最短路径，就是源点构成的字符串。

(3)最短路径上的某个顶点，可能有多个前置邻接点。

(4)从源点出发到某个顶点，可能存在多条最短路径。

(5)某个顶点，根据前置邻接点的最短路径，连接上本顶点，就构成了本顶点的最短路径。

4.3　数据结构设计

根据上述对多条最短路径问题的分析，设计数据结构。

(1)对所有 N 个顶点进行编号：$0,1,\cdots,N-1$。

起始点 start 编号为0。

(2)字符串数组 sVertexName[N]，表示各顶点的名称。数组下标为各顶点对应的编号(下同，略)。

(3)整数数组 iPreVertexNum[N]，表示各顶点前置邻接点的数量(可能有多个前置邻接点，最多为 $N-1$)。

起始点 start 没有前置邻接点。

所有顶点前置邻接点的数量初始化为0。

iPreVertexNum[i]=0($i=0,1,\cdots,N-1$)。

(4)整数数组 iPreVertexID[N][N]，表示各顶点前置邻接点的编号(某顶点可能有多个前置邻接点，对应多个编号)。

(5)整数数组 iMinPathNum[N]，表示(从起始点到)各顶点的最短路径的数量(有可能大于 N)。

初始化：

iMinPathNum[i]=0($i=0,1,\cdots,N-1$)。

iMinPathNum[0]=1，表示从起始顶点 start 到自身，最短路径的数量为1。

(6)字符串数组 sMinPath[N][M]，表示(从起始点到)各顶点的所有最短路径(设置 M 的值大于 N)。

初始化：

sMinPath[i][j]=""($i=0,1,\cdots,N-1$；$j=0,1,\cdots,M-1$)。

sMinPath[0][0]= "start"，表示从起始点 start 到自身的最短路径只有一条，是起始点自身构成的字符串。

4.4　算法设计

根据上述数据结构设计，对 Dijkstra 算法进行改进，设计算法，求取(从起始点到)各顶点的多条最短路径，以及每个顶点所有的前序邻接点。

Step 1　初始化。

对所有顶点编号，构造邻接矩阵。

构造并初始化(从源点到)各顶点的最短路径及个数。

构造并初始化各顶点前面的邻接点及个数。

给源点设定永久性标号(p 标号)，其余顶点都不是永久性编号(p 标号)。

根据源点，求其余各顶点(源点的邻接点)的：最短路径、长度、数量；前面邻接点及其个数。

Step 2　求下一个永久性标号(p 标号)顶点。

通过邻接矩阵，找到没有永久性编号(p 标号)的、其最短路径长度最小的顶点 v。

如果这个顶点 v 的最短路径长度为无穷大，那么退出；否则，给这个顶点 v 设定永久性编号(p 标号)。

根据这个顶点 v，以及邻接矩阵，求其余各顶点(顶点 v 的邻接点)的：最短路径、长度、数量；前面邻接点及其个数。

如果源点经过顶点 v 到其余各顶点 x(顶点 v 的邻接点)的路径长度小于顶点 x 现有的最短路径长度，那么：

顶点 x 最短路径长度为经过顶点 v 到顶点 x 路径长度。

顶点 x 的前置顶点为 v，顶点 x 的前置顶点个数为1。

顶点 x 最短路径个数，为顶点 v 的最短路径个数。

顶点 x 的最短路径，为顶点 v 的每条最短路径连接顶点 x 所构成的字符串。

如果源点经过顶点 v 到其余各顶点 x(顶点 v 的邻接点)的路径长度等于顶点 x 现有的最短路径长度，那么：

顶点 x 的前置顶点包含 v，顶点 x 的前置顶点个数加1。

顶点 x 最短路径个数，为顶点 x 现有最短路径个数与经过顶点 v 的最短路径个数之和。

将顶点 v 的每条最短路径连接顶点 x 所构成的字符串作为一条新的最短路径，加入顶点 x 的最短路径集合中。

Step 3　求下一个永久性标号(p 标号)顶点。

如果所有顶点都有永久性标号(p 标号)，那么退出程序。

否则转 Step 2。

4.5　算法伪码

```
1. function Dijkstra(Graph,source):
2.     code each vertex with an ID
3.     create adj[N][N] according to Graph
4.     create sVertexName[N] according to Graph
5.     create iMinPathNum[N] and sMinPath[N][M]
6.     create iPreVertexNum[N] and iPreVertexID[N][N]
7.     for each vertex v in Graph:
8.         final[v] := false;
9.         dist[v] := adj[source][v];
10.        iPreVertexNum[v] := 0;
11.        iMinPathNum[v] := 0;
12.    if(v is not source and dist[v] is not Infinite)
13.        iMinPathNum[v] := 1;
```

```
14.          sMinPath[v][0] := source name add v name;
15.          iPreVertexNum[v] := 1;
16.          iPreVertexID[v][0] := source;
17.      end if
18.          final[source] := true;
19.          iMinPathNum[source] := 1;
20.          sMinPath[source][0] := name of source;
21.          iPreVertexNum[source] := 0;
22.      end for
23.      Loop N-1 times:
24.      find smallest dist[v] in dist[] and ! final[v];
25.          if dist[v]=infinity:
26.            break;
27.          end if
28.          final[v] := true;
29.          for each vertex w in Graph:
30.      if(!final[w]&& dist[v]+adj[v][w]< dist[w])
31.                  dist[w] :=dist[v]+adj[v][w];
32.                  iMinPathNum[w] := iMinPathNum[v];
33.              replace sMinPath[w] with sMinPath[v];
34.                  each sMinPath[w] add name of w;
35.                  iPreVertexNum[w] :=1;
36.                  iPreVertexID[w][0] :=v;
37.              end if
38. else if(!final[w]&& dist[v]+adj[v][w]==dist[w])
39.                  iMinPathNum[w] plus iMinPathNum[v];
40.                  sMinPath[w] plus sMinPath[v];
41.                  iPreVertexNum[w] plus 1;
42.                  iPreVertexID[w] plus v's ID;
43.              end else if
44.          end for
45.      end Loop
46.      return:dist
47.      return:iMinPathNum[N] and sMinPath[N][M]
48. return:iPreVertexNum[N] and iPreVertexID[N][N]
49. end function
```

4.6 算法的时间复杂度

本算法是 Dijkstra 算法的改进，通俗来讲，是严格依附于 Dijkstra 算法的，所以本算法在时间复杂度上与 Dijkstra 算法是一致的。

本算法的运行时间表示为边数 m 和顶点数 n 的函数。Dijkstra 算法最简单的实现方法是用一个链表或者数组来存储所有顶点的集合 Q，所以搜索 Q 中最小元素的运算（Extract-Min(Q)），只需要线性搜索 Q 中的所有元素。这样，算法的运行时间是 $O(n^2)$。

当用到二叉堆的时候，算法所需时间为 $O((m+n)\log n)$，斐波纳契堆能稍微提高一些性能，让算法运行时间达到 $O(m+n\log n)$。

5 算法实验

用上述算法进行实验。实验环境是 MacBook Air 笔记本电脑：处理器 1.3GHz Intel Core i5，内存 4GB 1600MHz DDR3，启动硬盘 Macintosh HD；OS X 10.8.4 (12E3067)操作系统。

5.1 算法实验 1

图 5 是一个较为典型的求取多个邻接点与多条最短路径的例子。从 V_0 点出发，到 V_6 点有 5 条最短路径，到 V_9 点有 15 条最短路径。V_6 点前面有 5 个邻接点，V_9 点前面有 3 个邻接点。

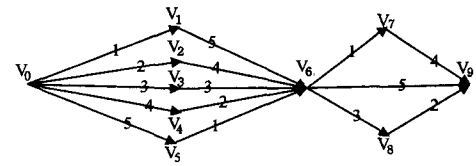

图 5 有向带权图 G3

5.2 实验结果 1

本程序运行时间 489ms。

程序运行结果如下(节选其中的多条最短路径结果)：

from source V_0 to vertex V_6, the number of shortest path is:5, list all the shortest path:

$V_0V_1V_6$; $V_0V_2V_6$; $V_0V_3V_6$; $V_0V_4V_6$; $V_0V_5V_6$

from source V_0 to vertex V_7, the number of shortest path is:5, list all the shortest path:

$V_0V_1V_6V_7$; $V_0V_2V_6V_7$; $V_0V_3V_6V_7$; $V_0V_4V_6V_7$; $V_0V_5V_6V_7$

from source V_0 to vertex V_8, the number of shortest path is:5, list all the shortest path:

$V_0V_1V_6V_8$; $V_0V_2V_6V_8$; $V_0V_3V_6V_8$; $V_0V_4V_6V_8$; $V_0V_5V_6V_8$

from source V_0 to vertex V_9, the number of shortest path is:15, list all the shortest path:

$V_0V_1V_6V_9$; $V_0V_2V_6V_9$; $V_0V_3V_6V_9$; $V_0V_4V_6V_9$; $V_0V_5V_6V_9$; $V_0V_1V_6V_7V_9$; $V_0V_2V_6V_7V_9$; $V_0V_3V_6V_7V_9$; $V_0V_4V_6V_7V_9$; $V_0V_5V_6V_7V_9$; $V_0V_1V_6V_8V_9$; $V_0V_2V_6V_8V_9$; $V_0V_3V_6V_8V_9$; $V_0V_4V_6V_8V_9$; $V_0V_5V_6V_8V_9$

cost time:489 ms

5.3 算法实验 2

图 6 是福建莆田市 7 个主要景点旅游线路的优化问题，这 7 个景点之间的关系可用一个加权无向图 G 来表示。可以描述为：从莆田市古谯楼(市中心)(顶点 1)出发，到达各个景点的最短路径。这个例子有一定的实际意义。

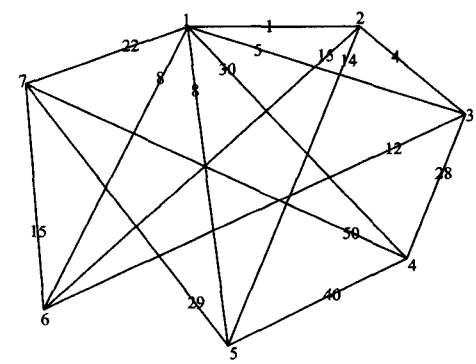

图 6 无向带权图 G4

5.4 实验结果 2

本程序运行时间 357ms。

程序运行结果如下：

from source V_1 to vertex V_3, the number of shortest path is:2, list all the shortest path:

V_1V_3; $V_1V_2V_3$

cost time:357 ms

5.5 实验结果分析

从实验结果可以看出，本算法可以正确处理多邻接点问题和多条最短路径问题。

6 用布尔邻接(可达)矩阵验证实验结果

下面通过邻接矩阵从侧面验证上述实验结果。首先给出邻接矩阵的定义：

设简单图 $G=\langle V,E\rangle$，$V=\{V_1,V_2,\cdots,V_n\}$，$|E|=m$，称 n 阶方阵 $A(G)=(a_{ij})$ 为有向图 G 的邻接矩阵(Adjacency Matrix)。其中，

$$a_{ij}=\begin{cases}1, & v_i \text{ 邻接 } v_j\\ 0, & \text{不邻接 } v_j \text{ 或 } i=j\end{cases}$$

邻接矩阵中，其元素非 0 即 1，称这种 0-1 矩阵为布尔矩阵。通过图的邻接矩阵可以得到图的很多重要性质：

(1)当给定的简单图是无向图时，邻接矩阵为对称的；当给定的图是有向图时，邻接矩阵不一定对称。

(2)对有向图来说，邻接矩阵 $A(D)$ 的第 i 行中 1 的个数是顶点 V_i 的出度，第 j 列中 1 的个数是顶点 V_j 的入度。

(3)零图的邻接矩阵的元素全为零，叫做零矩阵；反过来，如果一个图的邻接矩阵是零矩阵，则此图一定是零图。

(4)如果邻接矩阵 A 的 (i,i) 项元素(第 i 行，第 i 列)为 1，说明顶点 V_i 上有环。

下面给出一个定理：

定理 1 设 D 是具有 n 个顶点 $\{V_1,V_2,\cdots,V_n\}$ 的图，其邻接矩阵为 A，则 $A_k(k=1,2,\cdots)$ 中的 (i,j)(第 i 行，第 j 列)项元素 $a_{ij}^{(k)}$ 表示从顶点 V_i 到顶点 V_j 的长度等于 k 的路的总数。

下面通过这个定理，对上述的实验结果进行验证。

6.1 实验 1 结果验证

实验 1 中的有向图 G3，布尔邻接矩阵 $A(G3)$、$A^2(G3)$、$A^3(G3)$、$A^4(G3)$ 分别如图 7 所示。

$$A(G3)=\begin{bmatrix}0&1&1&1&1&1&0&0&0&0\\0&0&0&0&0&0&1&0&0&0\\0&0&0&0&0&0&1&0&0&0\\0&0&0&0&0&0&1&0&0&0\\0&0&0&0&0&0&1&0&0&0\\0&0&0&0&0&0&1&0&0&0\\0&0&0&0&0&0&0&1&1&1\\0&0&0&0&0&0&0&0&0&1\\0&0&0&0&0&0&0&0&0&1\\0&0&0&0&0&0&0&0&0&0\end{bmatrix}\quad A^2(G3)=\begin{bmatrix}0&0&0&0&0&0&5&0&0&0\\0&0&0&0&0&0&0&1&1&1\\0&0&0&0&0&0&0&1&1&1\\0&0&0&0&0&0&0&1&1&1\\0&0&0&0&0&0&0&1&1&1\\0&0&0&0&0&0&0&1&1&1\\0&0&0&0&0&0&0&0&0&2\\0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0\end{bmatrix}$$

$$A^3(G3)=\begin{bmatrix}0&0&0&0&0&0&0&5&5&5\\0&0&0&0&0&0&0&0&0&2\\0&0&0&0&0&0&0&0&0&2\\0&0&0&0&0&0&0&0&0&2\\0&0&0&0&0&0&0&0&0&2\\0&0&0&0&0&0&0&0&0&2\\0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0\end{bmatrix}\quad A^4(G3)=\begin{bmatrix}0&0&0&0&0&0&0&0&0&10\\0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0\end{bmatrix}$$

图 7 布尔邻接矩阵 $A(G3)$、$A^2(G3)$、$A^3(G3)$、$A^4(G3)$

从上述布尔邻接矩阵可以得到如下结论：

(1)$a_{12}^{(1)}=1$，$a_{13}^{(1)}=1$，$a_{14}^{(1)}=1$，$a_{15}^{(1)}=1$，$a_{16}^{(1)}=1$，所以 V_0 到 V_1，V_0 到 V_2，V_0 到 V_3，V_0 到 V_4，V_0 到 V_5，经过 1 条边的所有路径，分别有 1 条。

(2)$a_{17}^{(2)}=5$，所以 V_0 到 V_6，经过 2 条边的路径，共有 5 条。

(3)$a_{18}^{(3)}=5$，$a_{19}^{(3)}=5$，$a_{110}^{(3)}=5$，所以 V_0 到 V_7，V_0 到 V_8，V_0 到 V_9，经过 3 条边的所有路径，分别有 5 条。

(4)$a_{110}^{(4)}=10$，所以 V_0 到 V_9，经过 4 条边的路径，共 10 条。

这与上述实验 1 的结果是完全吻合的。

6.2 实验 2 结果验证

实验 2 中的无向图 G4，布尔邻接矩阵 $A(G4)$、$A^2(G4)$、$A^3(G4)$、$A^4(G4)$ 分别如图 8 所示。

$$A(G4)=\begin{bmatrix}0&1&1&1&1&1&1\\1&0&1&0&1&1&0\\1&1&0&1&0&1&0\\1&0&1&0&1&0&1\\1&1&0&1&0&0&1\\1&1&1&0&0&0&1\\1&0&0&1&1&1&0\end{bmatrix}\quad A^2(G4)=\begin{bmatrix}6&3&3&3&3&3&3\\3&4&2&3&1&2&3\\3&2&4&1&3&2&3\\3&3&1&4&2&3&2\\3&1&3&2&4&3&2\\3&2&2&3&3&4&1\\3&3&3&2&2&1&4\end{bmatrix}$$

$$A^3(G4)=\begin{bmatrix}18&15&15&15&15&15&15\\15&8&12&9&13&12&9\\15&12&8&13&9&12&9\\15&9&13&8&12&9&12\\15&13&9&12&8&9&12\\15&12&12&9&9&8&13\\15&9&9&12&12&13&8\end{bmatrix}\quad A^4(G4)=\begin{bmatrix}90&63&63&63&63&63&63\\63&52&44&49&41&44&49\\63&44&52&41&49&44&49\\63&49&41&52&44&49&44\\63&41&49&44&52&49&44\\63&44&44&49&49&52&41\\63&49&49&44&44&41&52\end{bmatrix}$$

图 8 布尔邻接矩阵 $A(G4)$、$A^2(G4)$、$A^3(G4)$、$A^4(G4)$

从上述布尔邻接矩阵可以得到如下结论：

(1)$a_{12}^{(1)}=1$，$a_{13}^{(1)}=1$，$a_{14}^{(1)}=1$，$a_{15}^{(1)}=1$，$a_{16}^{(1)}=1$，$a_{17}^{(1)}=1$，所以 V_1 到 V_2、V_3、V_4、V_5、V_6、V_7，经过 1 条边的所有路径，分别有 1 条。

(2)$a_{13}^{(2)}=3$，所以 V_1 到 V_3，经过 2 条边的路径，共有 3 条(其中有一条是最短路径)。

6.3 验证结论

上面从布尔邻接(可达)矩阵的角度，对多条最短路径算法的实验结果进行了验证。

在连通图上(有向图或者无向图)，从某一点出发到达另外一点可能有多条路径，每条路径经过的边的个数有可能不同，但是最短路径的个数肯定少于两点之间所有路径的个数。这就是从布尔邻接(可达)矩阵的角度所要验证的。

7 应用研究

本文研究了 Dijkstra 算法中的多邻接点和多条最短路径问题，对多邻接点问题和多条最短路径问题进行了深入的分析，为解决多邻接点问题和多条最短路径问题，设计了数据结构和算法，给出了算法伪码并编程实现。

算法实验结果表明，改进之后的 Dijkstra 算法可以有效解决多邻接点问题和多条最短路径问题。通过邻接矩阵对实验结果进行了验证，从一个侧面说明实验结果是正确的。

7.1 应用于社会网络研究中

试图将本算法应用于社会网络中的最短路径选择，特别是处理多邻接点和多条最短路径问题。

社会网络(Social network)，是由许多节点构成的一种社会结构，节点通常是指个人或组织，社会网络代表各种社会关系，经由这些社会关系，把从偶然相识的泛泛之交到紧密结合的家庭关系的各种人们或组织串连起来。由此产生的图形结构往往是非常复杂的。

1954 年，J. A. Barnes 开始使用"Human Relations"这个术语，系统化地呈现关系模式，统整了大众与社会科学家的传统概念：有限制的群体(例如，部落，家庭)和社会分类(例如，性别，种族)。一个社交网络最大约为 150 人左右 (Dunbar's

number)，平均大小约为 124 人左右(http://zh.wikipedia.org/wiki/社会网络)。

社会网络分析已经成为一个关键技术，也是一项热门的研究，已应用于现代社会学、人类学、社会语言学、地理、社会心理学、通讯研究、资讯科学、社会网络分析与探勘、组织研究、经济学，以及生物学领域。一个多世纪以来，人们使用社交网络来比喻复杂的社交系统下成员之间的关系，囊括所有层级，从人际关系到国际关系。

社交网络在很多层面运作，从家庭到国家层面都有，并扮演着关键作用，决定问题如何得到解决，组织如何运行，并在某种程度上决定个人能否成功实现目标。用最简单的形式来说，社会网络是一张地图，标示出所有与节点相关的连结。社交网络也可以用来衡量个人参与者的社会资本。这些概念往往显示在一张社会网络图上，其中节点是点状，连结是线状。社会网络分析用来检视节点、连结之间的社会关系。节点是网络里的个人参与者，连结则是参与者之间的关系。节点之间可以有很多种连结。

可以将本算法应用于社会网络中，找到点与点之间的最短路径及其长度、最短路径的个数、最短路径上每个顶点的所有前序邻接点。这将有助于通过多条最短路径在点与点之间建立联系，从而达到节约成本的目的。

7.2 应用于 XBRL 的研究中

下一步尝试将本算法用于 XBRL 的研究中。

可扩展商业报告语言(eXtensible Business Reporting Language，XBRL)是一种基于 XML 的标记语言，用于商业和财务信息的定义和交换。以这种语言为基础，通过对网络财务报告信息的标准化处理，可以编制出比现行网络财务报告更加先进的报告，可以将网络财务报告的不能自动读取的信息转换为一种可以自动读取的信息，大大地方便信息使用者对信息的批量需要和批量利用。XBRL 具有低信息交换成本、高信息相关性、普适性等优点。但由于国内的特殊背景，XBRL 在国内一直没有得到普及。

对 XBRL 优势的研究，可从管制者、证券交易所、税务当局、分析师、投资者、债权人以及报告编制者等财务报告供应链上各参与方的角度考察。具体表现在：

(1)对管理者：可以收集更多的数据，不同机构间分享标准、准确、可重复使用的数据定义；因为 XBRL 提供了数据类型与数据结构验证，允许文档在发送前验证，所以数据更准确；XBRL 应用软件可以独立于某一个分类标准，这个特性支持报告要求不断随企业情况变化而变化；管制者可以控制遵从成本，通过促进 XBRL 的广泛使用，帮助受管制组织与其他利益相关者权衡其成本。

(2)对交易所：XBRL 提供了一个国际认可的信息披露格式，这将减少获取与分发数据的成本；提供了公司、交易所、管制者与分析师共享的数据定义，从而使得上市公司更可能实现国际披露；交易拥有认可不同会计与报告定义的能力。

(3)对税务当局：可以为报告者与税务当局节约成本和改善报告；使之更方便地获取相关信息进行分析，更快速地搜索到相关信息；可以减少纳税人的遵从成本。

(4)对分析师、投资者与债权人：当 XBRL 广泛使用后，分析师、投资者与债权人可利用智能搜索引擎直接从单个公司信息系统或网站收集相关信息进行比较；企业可以生产出更及时、更准确与可比较的财务报告。

(5)对报告编制者：显著提高利用 XBRL 技术获取信息与整合信息的能力，通过提高企业财务报告信息透明度，XBRL 技术有助于人们决策。许多研究结果表明，XBRL 在信息获取方面的确存在巨大优势。

从 2000 年 XBRL 研发取得初步成果至今，XBRL 国际已认可包括加拿大、德国、韩国、新西兰、英国等多个国家以及 IASB、GAAP 等国际组织的财务报告分类标准。这为 XBRL 的国际化和标准化奠定了基础。

XBRL 技术的应用提高了财务信息质量与透明度，方便数据的查询、提取和系统化管理，节省报送成本，提高监管效率，同时增加了监管机构监测风险的工具，但是否能发挥监测系统性风险、稳定金融系统的作用，还要取决于监管机构和政府决策者在数据分析、判断和决策上工作的质量。

从应用主体来看，XBRL 应用的最大支持者和受益者还是“上游”的政府机构，相对而言，“下游”终端用户对 XBRL 的商业应用还处于探索阶段。监管机构是推动 XBRL 技术应用及发展的主要动力，数据应用框架的主体仍然是“上游”的监管机构，他们不仅是需求者，系统建设的推动者，也是数据应用框架的构建者。但是，XBRL 数据的商业应用对数据的准确性、及时性和全面性提出了更高的要求，并反过来推动有关报送主体提高报送质量，改善治理结构，最终起到约束其经营行为，推动资本市场发展的作用。

应该指出，XBRL 是建立在 XML 编程语言之上的。所以，XML 数据的路径表达式查询优化技术是许多研究的重点。这方面的研究很多，例如：吕建华[18]研究了 XML 数据的路径表达式查询优化技术；孔令波[19]研究了 XML 数据的查询技术；孟小峰[20]研究了 XML 查询优化问题；孔令波[21]研究了 XML 数据索引技术。上述研究都指出了同一个问题：多个数据标签之间的最短路径问题。我们试图通过本算法，在 XBRL 的各种标签(tag)组成的树中找到标签之间的多条最短路径。这将是下一步的工作，也是非常有意义的一项工作。

结束语 根据文中的伪码，可以快速写出程序代码。本文写作的初衷之一，是因为在《计算机科学》上发表了论文《改进的 Dijkstra 最短路径算法及其应用研究》[2]，很多国内同行用各种方式表示了对本工作的关注，并诚恳地指出文中的一些错误。我们对同行的关注与热情帮助表示感谢，完全接受同行的批评并在本文中进行改正，并真诚地希望有更多的同行研究 Dijkstra 算法，研究最短路径算法，提高算法的性能，使之更加广泛地应用在各个领域。

参 考 文 献

[1] Dijkstra E W. A note on two problems in connexion with graphs [J]. Numberische Mathematik, 1959, 1(1): 269-271

[2] 王树西，吴政学. 改进的 Dijkstra 最短路径算法及其应用研究[J]. 计算机科学，2012，39(5)：151-153

[3] 董俊，黄传河. 改进 Dijkstra 算法在 GIS 导航应用中最短路径搜索研究[J]. 计算机科学，2012，39(10)：125-130

[4] 刘国华. 基于 Dijkstra 距离的聚类算法研究及其在物流中的应用[D]. 兰州：兰州大学，2011

[5] 郑四发,曹剑东,连小珉. 复杂路网下多客户间最短路径的扇面Dijkstra算法[J]. 清华大学学报:自然科学版,2009(11):115-120

[6] 刘建美,马寿峰,马帅奇. 基于改进的Dijkstra算法的动态最短路计算方法[J]. 系统工程理论与实践,2011(6):200-206

[7] 李敬贤,厉小润. 求解震后最优路径的改进Dijkstra算法[J]. 计算机工程,2012(3):177-183

[8] 周竞文,程志全,金士尧. 基于Dijkstra距离剪枝的测地线求解算法[J]. 系统仿真学报,2009(10):92-98

[9] 吴若伟,楼佩煌. 基于Dijkstra算法的大型停车场最优泊车路径规划[J]. 工业控制计算机,2013(5):183-189

[10] Idwan S, Etaiwi W. Dijkstra algorithm heuristic approach for large graph[J]. J Appl Sci,2011,12:2255-2259

[11] Medeiros F L L,da Silva J D S. A Dijkstra Algorithm for Fixed-Wing UAV Motion Planning Based on Terrain Elevation[J]. Lecture notes in computer science,2010,6404:213-222

[12] Gunkel C, Stepper A, Muller A C ,et al. Micro crack detection with Dijkstra's shortest path algorithm[J]. Machine Vision and Applications,2012,23(3):589-601

[13] Li R. Utilizing Restricted Direction Strategy and Binary Heap Technology to Optimize Dijkstra Algorithm in WebGIS[J]. Key Engineering Materials,2010(419/420):557-560

[14] Tintor V, Radunovi A J. Distributed Dijkstra sparse placement routing algorithm for translucent optical networks[J]. Photonic Network Communication,2009,18(1):55-64

[15] Kimoto M, Tsuchiya T, Kikuno T. On the Time Complexity of Dijkstra's Three-State Mutual Exclusion Algorithm[J]. IEICE Transactions on Information and Systems, 2009, 92(8): 1570-1573

[16] Bauer R, Delling D, Sanders P, et al. Combining Hierarchical and Goal-Directed Speed-Up Techniques for Dijkstra's Algorithm [J]. Lecture Notes in Computer Science,2008,5038:303-318

[17] 耿素云. 离散数学[M]. 北京:清华大学出版社,2008

[18] 吕建华,王国仁,于戈. XML数据的路径表达式查询优化技术[J]. 软件学报,2003(9):1193-1199

[19] 孔令波,唐世渭,杨冬青,等. XML数据的查询技术[J]. 软件学报,2007(6):1400-1418

[20] 孟小峰,王宇,王小锋. XML查询优化研究[J]. 软件学报,2006(10):2069-2086

[21] 孔令波,唐世渭,杨冬青,等. XML数据索引技术[J]. 软件学报,2005(12):1000-1017

(上接第216页)

表明改进后的算法对垃圾邮件的识别能力有所降低,但是算法改进后的召回率变化与算法改进前召回率的变化比较接近。由图2可以看出,在改进后的算法中,正确率有明显的提高,而且正确率的变化比较稳定,表明改进后的算法中过滤器对合法邮件的误判数量在减少,对合法邮件的误判率在降低,从而也降低了过滤器误判给用户带来的损失。由图3和图4可以看出,在改进后的算法中,过滤器的精确率在上升,错误率在下降,表明算法改进后过滤器的分类精度在提高。

结束语 本文针对垃圾邮件过滤中的特征项选择问题,利用特征词的先验概率定义增益比,对特征词的类别条件熵的计算做了改进,利用增益比对整个分类所提供的信息量进行放大或弱化,从而提出了一种改进的信息增益方法来提取特征词。最后在英文语料库上进行实验,实验中采用了极大后验假设的朴素贝叶斯决策方法,实验结果表明改进后的算法虽然漏掉了一部分垃圾邮件,但是合法邮件误判率在降低,对合法邮件判断更加准确,从而降低了对合法邮件的误判给用户带来的损失。

本文下一步研究的工作是在提高过滤器的正确率,降低用户损失的基础上,提高过滤器的召回率。

参 考 文 献

[1] Guzella T S, Caminhas W M. A review of machine learning approaches to spam filtering[J]. Expert Systems with Application, 2009,36(7):10206-10222

[2] Lai Chih-chin. An Empirical Study of Three Machine Learning Methods for Spam Filtering [J]. Knowledge-Based System, 2007,20(3):249-254

[3] 黄国伟,许昱玮. 基于用户反馈的混合型垃圾邮件过滤方法[J]. 计算机应用,2013,33(7):1861-1865

[4] 邓维斌,王国胤,洪智勇. 基于粗糙集的加权朴素贝叶斯邮件过滤方法[J]. 计算机科学,2011,38(2):218-221

[5] Sanchez F, Duan Zhen-hai, Dong Ying-fei. Understanding Forgery Properties of Spam Delivery Paths[C]// CEAS 2010 Seventh annual Collaboration, Electronic messaging, AntiAbuse and Spam Conference (CEAS 2010). Redmond, Washington, US, July 2010

[6] 陈孝礼,刘培玉. 应用于垃圾邮件过滤的词序列核[J]. 计算机应用,2011,31(3):698-701

[7] Sahami M, Dumais S, Heckerman D, et al. A Bayesian approach to filtering Junk e-mail [C]// Learning for Text Categorization: Papers from AAAI Workshop. Madison, Wisconsin, 1998:55-62

[8] Androutsopoulos I, Koutsias J, Chandrinos K V, et al. An Evaluation of Naive Bayesian Anti-Spam Filtering[C]// Proc of the Workshop on Machine learning in the New Information Age, 11th European Conference on Machine Learning (ECML'00). Barcelona, Spain, June 2000:9-17

[9] Schneider K. A Comparison of Event Models for Naive Bayes Anti-spam E-mail Filtering[C]// Procedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics(EACL'03). 2003:307-314

[10] Vangelis M, Androutsopoulos I, Georgios P. Spam filtering with Naive Bayes-which Naive Bayes? [C]// CEAS 2006 Third Conference on Email and AntiSpam(CEAS 2006). Mountain View, California USA, July 2006:27-28

[11] Chen Bin, Dong Shou-bin, Fang Wei-dong. Introduction of Fingerprint Vector based Bayesian Method for Spam Filtering [C]// CEAS 2007 Fourth Conference on Email and Anti-Spam(CEAS 2007). Mountain View, California USA, August 2007