计算机自动诊断中的数据挖掘问题

Data Mining Problems in Automatic Computer Diagnosis

（申请清华大学工学硕士学位论文）

培 养 单 位 ： 计算机科学与技术系

学　　科 ： 计算机科学与技术

研 究 生 ： 劳 逆

指 导 教 师 ： 李 春 平 副教授

二〇〇六年五月

计算机自动诊断中的数据挖掘问题

劳逆

关于学位论文使用授权的说明

本人完全了解清华大学有关保留、使用学位论文的规定，即：

清华大学拥有在著作权法规定范围内学位论文的使用权，其中包括：（1）已获学位的研究生必须按学校规定提交学位论文，学校可以采用影印、缩印或其他复制手段保存研究生上交的学位论文；（2）为教学和科研目的，学校可以将公开的学位论文作为资料在图书馆、资料室等场所供校内师生阅读，或在校园网上供校内师生浏览部分内容。

本人保证遵守上述规定。

（保密的论文在解密后遵守此规定）

作者签名：_____ 导师签名：_____

日　　期：_____ 日　　期：_____

摘　要

随着现代的计算机系统功能日趋强大，结构也越来越复杂，系统运行中出现越来越多的冲突和无法预料的结果。这些问题对于系统用户和专业维护人员都是一个重大的挑战。过去依靠人工诊断计算机问题的方法耗费大量的人力物力，而且对于复杂问题的诊断也愈加显得力不从心。因此来自计算机系统研究和数据挖掘两个方向的科学家都开始研究如何利用数据挖掘的方法帮助解决系统管理问题。他们提出了面向自动修复的计算（Recovery Oriented Computing）的概念。其中一个重要的问题是如何实现自动的错误诊断与修复。本文探索了利用多种系统动态或静态信息来识别已知问题的方法。

现有个人电脑上的诊断技术主要分为两种：一是通过高层次自然语言症状描述和信息检索系统帮助诊断，二是利用系统低层次的静态信息实现自动诊断。然而，第一种诊断方法仍然需要很多人力的投入，第二种诊断方法的精度还有待提高。本文率先提出将高层次自然语言症状描述和低层次的系统信息结合进行自动诊断的方法，并在大量真实数据的试验上证明了该方法良好的效果。本文还提出了结合这两种信息帮助检索文本知识库的四个概率模型（模型设计主要由微软亚洲研究院的文继荣研究员完成），并通过试验证明系统检索精度得到大幅度的提高。这一工作成为信息检索领域中首先成功的实现上下文查询并应用于实际问题的例子。

系统的动态信息有比静态信息更利于收集、应用范围更广、噪声更小等优点。但是由于其分析的复杂性，还没有被应用于个人电脑的诊断。现有的工作集中在计算机群和网络等较复杂的系统中。多是利用关联性分析、聚类、分类等数据挖掘方法帮助问题分析，却不能实现自动诊断。本文提出两层分类器的结构以实现完全无需人工干预的已知问题自动诊断。第一层分类通过错误症状的自动检测实现。第二层通过系统调用序列的分类来实现。这一方法应用在 Windows 系统的 5 个常见的问题上都取得了良好的效果。为了解决系统调用序列的分类问题，本文还提出了利用字符串比对(String Alignment)抽取序列数据分类特征的新方法。结合支持向量机（Support Vector Machine, SVM）分类器，该特征抽取方法明显优于传统的 n-gram 序列特征抽取方法。同时也优于隐马尔可夫模型（Hidden Markov Model, HMM）等经典序列分类方法。

关键词：自动诊断　注册表　上下文查询　排名算法　字符串对齐

Abstract

As the function and complexity of computer systems keep soaring, more conflictions and unexpected results come along. These problems become a serious challenge to both users and support professionals. Traditional troubleshooting methods relying heavily on human intervention make the process inefficient and the results inaccurate even for solved problems, which contribute significantly to user's dissatisfaction. Therefore, scientists from both system domain and data mining domain start to explore solving system management problems with data mining methods. The new conception of Recovery Oriented Computing (ROC) is increasingly becoming a hot research area. One of its important goals is to enable automatic identification of the root cause of a problem if it is a known one, which would further lead to its resolution. This thesis explored using various types of static and dynamic system information to build correlations with solved problems.

State-of-the-art diagnose techniques on personal computers can be roughly classified into two types: one, using high level natural language problem symptom description and Information Retrieval (IR) system to help system diagnosis; two, using low level system static information to achieve automatic diagnosis. However, the first approach still involves a lot of human intervention, and the second approach has unsatisfactory accuracy in many cases. This thesis takes the lead in combining both high level problem symptom description and low level system static information to achieve automatic diagnosis on personal computers. Its good accuracy is proved in experiment on many real life problems. This thesis also brings forward four probabilistic models which combine these two types of information to help retrieving troubleshooting documents from knowledge databases. Experiment results strongly support their effectiveness in improving retrieval accuracy. This work becomes the first in IR domain to build comprehensive models for Contextual Retrieval and prove its usefulness in practical problem.

Compared to system static information, dynamic information has many merits like easier for collection, wider application, and less noise information. However, because of its analysis complexity, it has not been used in diagnose techniques on personal computers. Most of its related works are concentrated on analyzing complex systems like computer clusters and networks. With the help of various data mining methods like clustering, classification, and association rule mining, dynamic information is used to facilitate hard problem diagnosis, but cannot achieve automated diagnosis. This thesis proposes a two level classifier to achieve fully automated diagnosis. The first level consists of automated symptom detection. The second level consists of classifiers built on system call sequences of know problem. In our experiment, the method achieves good accuracy in five common problems on Windows system. In order to solve the sequence classification problem, a new feature extraction method based on string alignment is proposed. Combined with Support Vector Machine (SVM) classifier, this method yields much better result than traditional n-gram sequence feature extraction, and also better than canonical sequence classification method like Hidden Markov Model (HMM).

Keywords: automatic diagnosis Windows registry
contextual retrieval ranking algorithm string alignment

目 录

第1章　引言

1.1　背景

现代的计算机系统功能日趋强大，结构也越来越复杂。这就产生了更多的系统冲突和无法预料的结果。过去依靠计算机系统专家诊断计算机问题的方法耗费着越来越大量的人力物力，而且对于复杂的问题也愈加显得力不从心。由此许多科学家开始研究利用数据挖掘的方法帮助解决系统管理问题，甚至实现自动的错误诊断。他们提出了面向自动修复的计算（Recovery Oriented Computing）的概念。应用数据挖掘的方法解决计算机系统诊断的问题也逐渐成为一个新兴的研究方向。

图灵奖获得者 Jim Gray [20]将未来免维护的系统（Trouble-Free System）作为 IT 业研究的重大课题：“一个每天有千千万万用户使用，却只需要一个人用业余时间来管理的系统”。为了实现这个目标，系统需要能够自我管理（self-managing）。Redstone 等[23]描绘了一个全球尺度的自动诊断系统。这个系统能自动从用户桌面收集信息，从全球的数据库中搜索问题的症状和解决方案。然而 Redstone 并没有能实现该系统。而其他的相关工作或者只能辅助诊断而不能实现自动诊断，或者自动诊断的精度不能完全满足实际需要。这将在第 2 章中详细叙述。

本文结合多种数据挖掘方法，先后从系统静态信息和动态信息两个方面入手实现了较少或完全无需人工干预条件下自动诊断已知问题的系统。

1.2　例子问题

为了了解计算机自动诊断问题是如何解决的这里首先需要对文中的几个常用概念作定义：

*症状（symptom）*是在问题发生过程中，用户能够体验到的一切事件。主要分为用户的行为和系统的响应。

*病因(root cause)*是导致问题发生的最根本原因。包括硬件系统、软件、数据文件、配置等等。例如“没有插上电源”就是“不能开机”这个问题的可能病因之一。

*问题(problem)*是由某病因造成用户体验到某*症状*的过程。

*解决方案(resolution)*是解决该问题的方法、程序、或者补丁等等。

这里给出一个典型的个人电脑问题的例子－ “IE 不能打开网页问题”。当用户在 Internet Explorer (IE)中打开一个网页的时候，他可能发现这个网页不能正确显示 (如图 1.1):

图 1.1　网页不能显示的问题

在 Microsoft 已知问题的知识库中查询我们发现这个问题有已知不止一个原因 (表 1.1)。

一个有经验的工程师需要将这些原因一一测试之后才能确定真正的原因是那一个。这就需要花费大量的时间。而且如果作为一个计算机系统的售后服务人员，很有可能还会需要亲临有问题的计算机现场才能做出诊断，这就更增加了售后服务的成本。因此实现自动的错误诊断与修复将极大的减少系统供应商的运营成本，具有很大的应用价值。而且对于用户来说，无需等待售后服务人员的问题诊断，及时解决当前面对的系统问题，也是最为迫切的需求。

表 1.1　IE 问题已知的原因

ID	Root Cause
1	Internet Explorer connection settings are incorrect.
2	An incorrect entry exists in a Hosts file.
3	The Winsock.dll, Wsock32.dll, or Wsock.vxd file is missing or damaged.
4	There are multiple copies of the Winsock.dll, Wsock32.dll, or Wsock.vxd file, or one of these files is in the wrong folder.
5	Transmission Control Protocol/Internet Protocol (TCP/IP) is not installed or is not functioning correctly.
6	The WinSock2 registry key is damaged.
7	The Internet Connection Sharing (ICS) installation is damaged or is not functioning.
8	The Rnr20.dll file is missing or damaged or the "Library Path" value in the following registry key is missing or contains the wrong location:
9	HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\WinSock2\Parameters\NameSpace_Catalog5\Catalog_Entries\000000000001
10	If you are using America Online (AOL) or CompuServe, there may be parental control restrictions applied to the account.
11	Incorrectly configured or non-functioning firewall or proxy software.
12	When you attempt to view web pages in Internet Explorer after you install MSN (Microsoft Network) 5.1, you may have similar symptoms.

1.3　一般自动诊断的流程

多数自动诊断系统都分为信息收集、问题分类、和解决方案反馈三个部分[26]。其工作流程一般分为一下五步（见图 1.2）：

1. 计算机出现问题；
2. 负责收集系统信息的程序将信息发送到诊断数据库；
3. 诊断数据库具有分类的能力，将该问题分到某个已知的原因；
4. 解决该问题的方法、程序、或者补丁被发送回有问题的计算机；
5. 计算机解决问题后恢复正常运行。

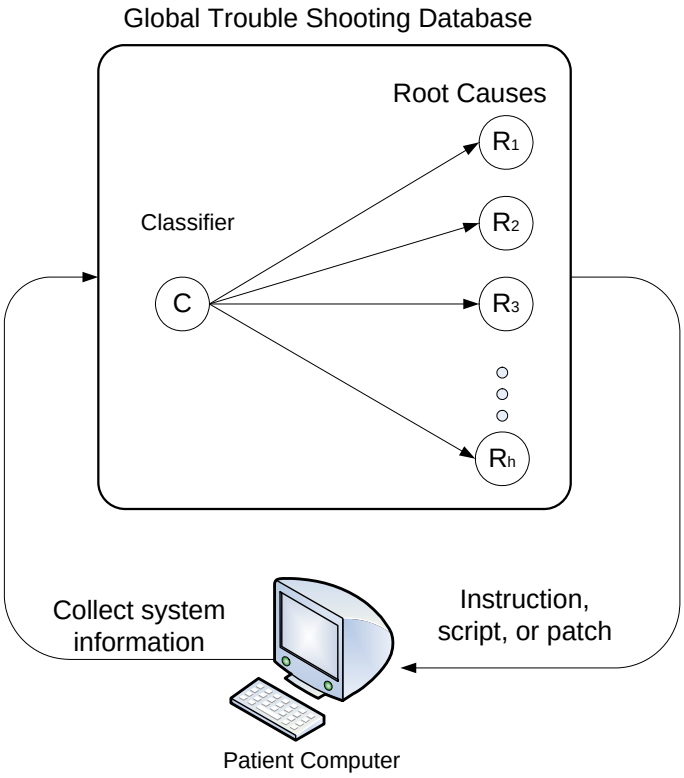

图 1.2　自动诊断的流程

本研究收集的系统信息包括两类: 静态信息和动态事件。

*静态信息*是指计算机固有的属性包括硬件系统、文件、配置信息等等。例如 Windows 系统中的注册表就是一种配置信息。

*动态事件*是指计算机在运行过程中产生的可以观测的现象。例如用户键盘鼠标的输入、产生窗口和菜单等用户界面事件。在系统底层的信息中还包括计算机系统函数的调用。

1.4　本文的主要方法和贡献

现有个人电脑上的诊断技术主要有两种：一是通过高层次自然语言症状描述和信息检索系统帮助诊断，二是利用系统低层次的静态信息实现自动诊断。然而，第一种方法仍然需要大量人力投入到诊断中，第二种方法的精度也还有待提高。由此本文首先提出结合现有技术中高层次自然语言症状描述和低层次的系统信息对可能的问题原因排名，进而实现自动诊断，并在大量真实数据的试验上证明了该方法良好的效果。

利用计算机基因库除了可以直接推测问题的原因，还可以帮助在知识库中查找问题相关的文献。这一功能可以帮助缺乏经验的工程师或者用户通过阅读问题相关文档了解问题。结合计算机基因库，本文又提出了帮助检索文本知识库的四个上下文查询概率模型，并通过试验证明系统检索精度得到大幅度的提高。这一工作成为信息检索领域中首先成功的实现上下文查询并应用于实际问题的例子。

为了利用系统的动态信息比静态信息更利于收集、应用范围更广、噪声更小等优点，本文提出结合错误症状的自动检测和系统调用序列的分类以实现完全无需人工干预的已知问题自动诊断。错误症状的自动检测通过 Windows 钩子(hook)技术[11]和 DLL 注射(injection)技术实现。为了解决系统调用序列的分类问题，本文还提出了利用字符串对齐抽取序列数据分类特征的新方法。结合支持向量机（Support Vector Machine, SVM）分类器，该特征抽取方法明显优于传统的 n-gram 序列特征抽取方法。同时也优于隐马尔可夫模型（Hidden Markov Model, HMM）等经典序列分类方法。将这一方法应用在 Windows 系统的 5 个常见的问题上都取得了良好的效果。

总结起来讲，本文的贡献主要在于以下四点：

1. 提出并实现了结合高层次的症状描述和低层次的系统信息来提高诊断精确度的方法；
2. 比较了四个上下文查询的概率模型；
3. 提出两层分类器的结构以实现完全无需人工干预的已知问题自动诊断。并且实现了自动检测问题症状的方法，以取代用户对症状的描述；
4. 提出新的基于字符串比对的序列数据分类特征抽取的方法，并给出了翔实的实验来证明其有效性。

1.5 论文的组织

本文接下来做如下安排：

第一章为引言，简单介绍文计算机自动诊断的研究意义、内容和一般方法，同时介绍本文的主要方法和贡献并罗列本文的章节安排。

第二章概括性地介绍计算机自动诊断的相关工作，包括计算机自动诊断、系统配置管理、通过数据挖掘的方法分析系统动态信息等。在介绍的

同时，分析了所提及算法的优缺点。

第三章介绍如何利用静态信息实现自动诊断系统。首先介绍如何结合高层次的症状描述和低层次的系统信息来提高诊断的精确度。在此基础上，本文提出并且比较了四个上下文查询的概率模型帮助文本知识库的检索。

第四章介绍如何利用动态事件实现自动诊断系统。首先介绍如何通过 Windows 钩子(hook)技术和 DLL 注射(injection)技术实现错误症状的自动检测。然后介绍了利用字符串对齐抽取序列数据分类特征的新方法。并结合 SVM 分类器将该方法同其他序列分类方法进行比较。

第五章总结全文，并展望文本分类的未来工作。

第2章 相关工作

本章将概括性地介绍计算机自动诊断的相关工作，包括计算机自动诊断、系统配置管理、通过数据挖掘的方法分析系统动态信息等。在介绍的同时，并详细分析了所提及算法的优缺点。

2.1 计算机系统的自动诊断

Redstone 等 [23]提出建立一个全球的包涵丰富系统信息的知识库将会成为实现自动诊断的一条途径。尽管他们提出了几种可能有用的系统信息，却没有给出如何能有效利用这些信息来实现自动诊断。Cohen 等[34]提出了一种能自动产生系统状态标识的方法，并且利用这些标识来问题。Kephart 和 Chess[35]提出与自动诊断相关的另一个概念自动计算（Autonomic Computing），目标是通过自动的实现检测、诊断和修复来使得系统拥有自我修复（self-healing）的能力。以上这些工作虽然规划了计算机系统的自动诊断未来的方向却都停留在设想阶段。并没有实现可以实用的系统。

Banga[36] 通过专家知识来设计系统，完成错误监测、组件完整性检测、跟踪系统配置变更等任务。这个方法的缺点是对每种运用软件或环境都需要相应的专家级知识。例如为了诊断 Outlook Express 的邮件数据文件损坏的问题，专家需要同Outlook Express开发人员一样拥有数据文件格式软件运行流程等知识。而这些知识，特别是在第三方软件中，是很难获得的。

本文结合多种数据挖掘方法，先后从系统静态信息和动态信息两个方面入手实现了较少或完全无需人工干预条件下能够自动诊断已知问题的系统。

2.2 系统配置管理中的自动诊断

“只要人们不断改变他们使用计算机系统的方式系统配置管理将一直是长期困扰使用者的问题” [17]。在如操作系统这样庞大而且拥有大量第三方软件的系统中，由于配置不当而造成的问题层出不穷而且一般也比较复杂[21].

目前工业界一般有两种不同的策略来诊断系统的错误：基于错误症状的方法和基于系统信息的方法。许多知名系统都拥有相关问题的在线知识数据库[①]（Knowledge Bases）。一般计算机用户多数只会使用基于错误症状的方法来解决配置产生的问题。他们用自然语言描述问题，然后用信息检索工具寻找相关该问题的文档由于多数用户都不是计算机专家，他们对问题的描述通常是不准确的。因此在信息检索中常常也得不到令人满意的结果。

在另一方面，许多工具利用系统的低层次信息来实现自动诊断[18][22]. 这些工具通常提供自己的语法用来描述系统的正常行为，用实时监控判断系统是否偏离正常的状态，并且定义一套处理行为集合来试图修复问题。例如 Strider [26]首先收集 Windows 系统中的注册表信息。然后通过一系列技术来缩小候选病因的集合，包括系统信息的比较(diff)、在线跟踪系统配置的访问、以及一些统计分析的方法。最后通过系统（配置）回卷[23]的方式来修复计算机系统。遗憾的是，在许多实际应用中这些方法的精确度还有待提高。

在第 3 章中，本文将介绍如何结合高层次的症状描述和低层次的系统信息来进一步提高诊断的精确度。

2.3 通过数据挖掘的方法分析系统动态信息

系统的动态信息有比静态信息更利于收集、应用范围更广、噪声更小等优点。但是由于其分析的复杂性，还没有被应用于个人电脑上的诊断中。现有的工作集中在计算机群和网路等较复杂的系统中利用关联性分析、聚类、分类等一系列数据挖掘方法，帮助问题分析。

Pinpoint [37][38]利用统计学习技术在 Web 服务计算机群（Web server farm）的环境下诊断系统故障。在收集到各个用户请求的记录后，数据挖掘的方法被用于同该故障最相关的系统组件。

Magpie[39][40] 的目标是分析并行的系统。标注了过去资源使用信息的行为模型是通过对过去事件记录的聚类来实现的。聚类的过程中的距离由记录的编

[①] Apple Corporation, AppleCare Knowledge Base, http://kbase.info.apple.com
BugNet, BugNet, http://www.bugnet.com.
Microsoft Corporation, Microsoft Knowledge Base, http://support.microsoft.com
Redhat Corporation, Redhat Support Forums, http://www.redhat.com/support/forums

辑距离来度量。这个模型可以帮助分析系统效率下降的瓶颈所在。

Aguilera 等[41] 寻求只通过结点间的来往信息记录来判断系统效率下降的瓶颈。这样就可以避免以上几种方法需要的应用层的分析。主要的方法是挖掘结点间事件的因果关系来判断系统效率下降的根本原因。

Cohen 等[42]通过训练分类器来分析系统效率问题和违反网络服务器规则的事件。分类器的输入是系统行为的统计信息，输出是某一规则是否被违反。

一些集中式的入侵检测系统[43][44][45] [46]（intrusion detection systems IDS)利用系统调用序列（system call sequences）的分类来识别入侵行为。这些方法是通过系统过去的行为记录训练分类器，然后对新的行为作判断。虽然目的不是自动诊断而是预警，方法上同本文是最为相近的。本文与之不同的是首先实现了自动检测问题症状的方法，以取代用户对症状的描述作为第一层分类器。然后本文提出新的基于字符串比对的序列数据分类特征抽取的方法，并给出了翔实的实验来证明其在第二层分类器中的有效性。

第3章 基于静态信息的诊断

本章将介绍如何结合高层次的症状描述和低层次的系统信息来提高诊断的精确度。基本的思想是从文本知识库中抽取高层次的症状描述和低层次的系统信息间的关联信息。然后用症状描述的相似度来对低层次的系统信息排名。这个方法应用在 Windows 注册表[19]产生的问题上。无论是潜在病因的排名还是帮助寻找问题相关文档都取得了良好的效果。最后本文通过数据分析了建立计算机基因库的优点所在。

3.1 基本原理

*静态信息*是指计算机系统中一切固有的信息，包括硬件系统、软件、数据文件、配置等等。任何计算机系统问题归根到底都是计算机的静态信息造成的。例如我们常常会有这样的经验：

“我的计算机昨天还好好的，今天就不能工作了!”

“他的计算机没有这个问题，为什么我的就有？”

“这台计算机的这个帐号没有问题，为什么那一个帐号会有？”

“我重启程序，重启机器，但是问题依然存在……”

每一个这样的问题后面都一定有静态信息的改变，不管是硬件系统、文件、还是配置信息。

如果我们能够扫描问题系统的静态信息，并检测出可能存在问题的信息，则不失为一种有效的诊断方法。就好像在显微镜下观察病人的血液，通过发现作为病原体的细菌或者病毒来诊断病情。

3.1.1 病因筛选过程

基于静态信息的诊断中静态信息的数量可能是巨大的。例如 Windows XP 系统一般有 20 万以上个注册表项，10 万以上的文件。这些静态信息中一般只有一个是问题的真正原因（也有少数问题是由多个静态信息共同引起的）。

如何从中筛选出有问题的信息呢？我们可以找到如下的规律：

1. 将问题系统的静态信息同正常系统的一一比较，不相同的那些才可能是问题的原因。

2. 在发生问题的过程中被访问到的信息才有可能是产生问题的原因。

利用这两条我们可以构建如图 3.1 的信息过滤系统[26]。程序在用户端首先要扫描问题计算机当前的注册表或文件等静态信息。以注册表信息为例，这些静态信息可以方便的通过 Windows API 访问得到，也可以直接读取 Windows 储存注册表信息的数据库文件得到。

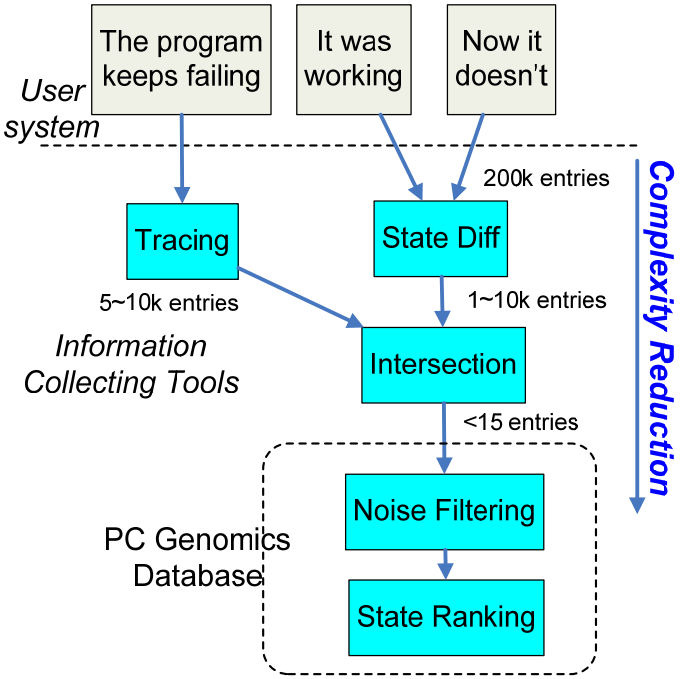

图 3.1 病因筛选过程

这些问题状态下的静态信息需要与一个正常计算机的静态信息做比较得出*比较集合(diff set)*。这个正常的静态信息可以是由 Windows 系统还原[2]功能保存的该计算机从前的信息，也可以从另外一台健康的计算机获得。

*跟踪集合(trace set)*是在出问题的过程中被访问到的静态信息。监测集合是通过 *Window 钩子技术* [11]，在系统核心态注入一个检测驱动程序检查所有系统调用的参数。由于应用程序访问任何资源都需要通过系统调用，所以钩子技术可以方便的检测所有系统信息被访问的情况。以注册表信息为例，每当某个应用程序通过 OpenReg 函数打开某一注册表项时，钩子程序则可以截获这一事

[2] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwxp/html/windowsxpsystemrestore.asp

件并根据其传入的函数参数得知请求访问的是哪一个注册表项。钩子程序在用户发生问题的过程中记录下所有程序的静态信息访问记录则成为跟踪集合(trace set)。

很显然，造成系统问题的信息一定在这跟踪集合集和比较集的*交集(intersection set)*里面。

试验证明这种过滤方法十分有效。初始的 20 万注册表集合通过比较之后一般只有 1 千至 1 万条;再同跟踪集合集取交后一般只有 15 条以内的注册表项。然而 15 对于一个实用的诊断系统来说还是比较大的。试想如果专家在诊断问题时还需要在 15 个可能原因中一个个排除，效率仍然是不能让人满意的。

本文提出将高层次自然语言症状描述和低层次的系统信息结合来进一步筛选这些信息，并且将它们按照出问题的概率排序，以提高自动诊断的精度。这一方法是通过建立一个*计算机基因库*（*PC-Genomics Database* 由 Ni 等[14]首次提出，而由本文首先实现），以及可能原因的排序算法实现的。计算机基因库中存储的是每个计算机静态信息及其对应已知引起过的问题文字描述。基因库的名字源于同人类基因库的类比。人类基因库中存储的是每个基因与其已知功能的对应。

如何建立计算机基因库以及可能原因的排序算法将在接下来一节中分别介绍。

3.1.2 系统结构以及应用场景

从文本知识库中抽取高层次的症状描述和低层次的系统信息间的关联信息被收集并发送到服务器。图 3.2 体现了计算机基因库如何利用这些信息帮助诊断的过程。

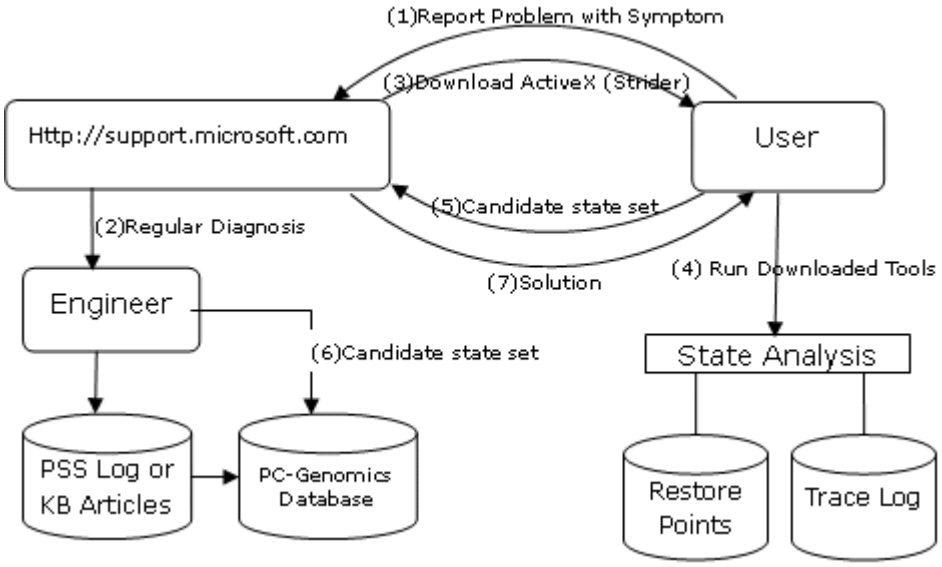

图 3.2 基于静态信息的诊断流程

例如一个叫黛安娜的用户“无法在字体对话框中找到任何字体”。这是因为注册表中关于 TrueType 的项目被损坏了。以下 7 个步骤说明了她的问题是怎样被解决的。

- **Step 1**: 黛安娜将问题报告到售后服务部门(例如 http://support.microsoft.com)
- **Step 2**: 工程师先采用传统的方法解决该问题。如果成功则转第 7。
- **Step 3**: 信息收集和分析工具被从服务器下载到黛安娜的电脑。
- **Step 4**: 黛安娜运行 Strider 工具来比较好的和坏的还原点信息，然后运行产生跟踪记录。
- **Step 5**: 可能的病因被发回到工程师。
- **Step 6**: 通过计算机基因库对可能的病因排名
- **Step 7**: 解决方案（可能是文章，可执行程序、或者软件组件）被送回黛安娜。在这个例子中，她将收到一段代码删除注册表中的 key_local_machine\software\microsoft\windows nt\currentversion\fonts。

传统的方法中所有用户端的信息都是通过用户电话或邮件发送给工程师，这不但延长了运行周期而且难以保证信息的准确性。在自动诊断系统中，多数用户端的信息是通过程序自动收集则避免了以上两个问题。

另外传统的方法完全依靠工程师的专业知识进行诊断，不但耗费人力而且

对人员的素质要求较高。通过建立计算机基因库，从前的诊断知识可以在遇到相同问题时得到再次利用。例如“升级某补丁失败”这样大面积发生的问题可能在上百万的用户中重复出现。传统的方法中每个这样的报修电话都需要工程师接听、诊断、处理消耗了大量不必要的人力。如果实现自动诊断系统则将产生可观的经济效益。

3.2 利用计算机基因库推测问题的原因

在过去的事件中于工程师以及用户们已经积累了大量诊断问题的经验，能否以某种方式将这些经验应用到一个自动诊断的系统中去呢？这正是建立计算机基因库的出发点。这一节将介绍如何构建计算机基因库来进一步筛选这些信息，并且将它们按照出问题的概率排序。

3.2.1 文字形式的知识源

我们发现这些知识大量的以文字的形式积累于网上数据库中。例如：

– AppleCare Knowledge Base, http://www.bugnet.com
– BugNet. http://www.bugnet.com
– Microsoft Knowledge Base, http://support.microsoft.com
– Redhat Support Forums, http://www.redhat.com/support/forums.

微软“知识库”（Knowledge Base, KB）包涵有经验的专家对已知微软产品问题的精确描述（如图 3.3）。一般分为标题、相关软件、问题症状、问题原因、和解决方法五个字段。这一数据源有很好的结构性、且内容可靠、描述准确。

微软“售后服务（Product Support Service, PSS）记录”保存了售后服务工程师解决每一个案例的详细过程（图 3.4）由解决问题的工程师书写。一般是电话或通信记录以及工程师最后做的总结。这一数据源有一定的结构性，但是可靠性和准确性都不如“知识库”。优点在于数量巨大问题涵盖面广。

另外有更多的知识存在与互联网上的论坛、个人网页等媒介中。但是由于其结构更加复杂，可靠性还不能保证所以不在本文的应用范围内。

Q329134: Print or Edit Dialog Boxes May Not Appear in Internet Explorer

The information in this article applies to:
Microsoft Internet Explorer version 6 for Windows 2000
Microsoft Internet Explorer 5.5 for Windows 2000 SP 2

SYMPTOMS
When you click Print or Print Preview on the File menu …

CAUSE
This problem occurs if a corrupted value exists in the registry that …

RESOLUTION

1. Click Start, click Run, type regedit in the Open box, and then click OK.
2. Locate and then click the following registry key:

HKEY_CLASSES_ROOT\CLSID\{00020420-0000-0000-C000-000000000046}\InprocServer32

3. In the right pane, right-click InprocServer32, and then click Delete.

图 3.3 微软“知识库”Knowledge Base

Case: 1234567
----------mail----------
Contact: Dina
System: WIN98 win 98 4.10
Problem: All of my true type fonts have vanished from the font dialog box
----------mail----------
Dear Gary,
…
----------mail----------
Dear Dina,
…
----------mail----------
SUMMARY
<<Symptom>>
TrueType fonts may not be present in the Fonts folder.
<<Cause>>
The registry key that lists TrueType fonts may be damaged or missing.
<<Resolution>>
Delete the Fonts key and then add it again under:
hkey_local_machine\software\microsoft\windows nt\currentversion

图 3.4 售后服务记录（Product Support Service Log）

3.2.2 构建计算机基因库

数字化的知识大量存在于自然语言书写的记录中。虽然我们远不能自动理解这些文字，但是通过*信息抽取*的方法我们能够识别文中的静态信息并且定位其所对应描述问题症状的文字。这种“静态信息－*症状*”的对应关系正是诊断知识的关键所在。

为了准确的从文本中识别静态信息的名称，我们首先建立了一个静态信息名称词典。以注册表为例，由于这些名称往往不但来自 Windows 系统本身，还来自每个第三方软件所以我们只能尽量多的从真实用户的电脑中采集。Windows 注册表有着树状的目录结构，每个结点上都可能存在多个配置。每个配置由配置名称和配置数据构成。所以在文本中若要唯一的匹配一个注册表项，则需要同时准确的找到其路径名和配置名。由于工程师在书写路径名的时候可能出现简写、插入叙述性文字、顺序颠倒等不规范情况，所以匹配算法在兼顾运行效率的同时也要实现一定的灵活性和智能性。

另外为了使用的方便和实现多用户的配置兼容问的注册表的目录结构还有许多重定向的目录。这使得不同的路径名可能对应于相同的配置，而相同的路径名可能对应于不同的配置。这些重定向规则在注册表的设计文档中均有说明。根据这些规则我们将相同注册表项*规范化*到一个唯一的名称。

试验中我们利用 50 台电脑的注册表建立了一个注册表名称字典。其中包涵 898,546 个不重复的注册表名称。利用这个字典，我们从 2,311,492 篇 PSS 记录中抽取到 143,157 个注册表名称，不重复的有 4,837 个。从 142,448 篇 KB 中抽取到 1,921 个注册表名称，不重复的有 996 个。

使用类似的方法，我们可以从文本中同时抽取出相关软件名称或版本号的信息。问题症状的描述，软件的名称，解决问题的步骤等信息在微软知识库中根据 xml 数据 tag 就可以识别。而在 PSS 记录中则需要对这些段落标识名称进行灵活的识别。

我们实现了从微软“知识库”和售后服务记录中抽取有用的诊断信息，并以之应用于静态信息的过滤和排序。在每篇文章中我们抽取造成问题的静态信息名称，问题症状的描述，软件的名称，解决问题的步骤（见图 3.5）等信息，然后储存到计算机基因库（PC Genomics Database）中。这一部分工作的的精度(precision)和识别率(recall)将直接影响到之后自动诊断算法的精度和识别率。

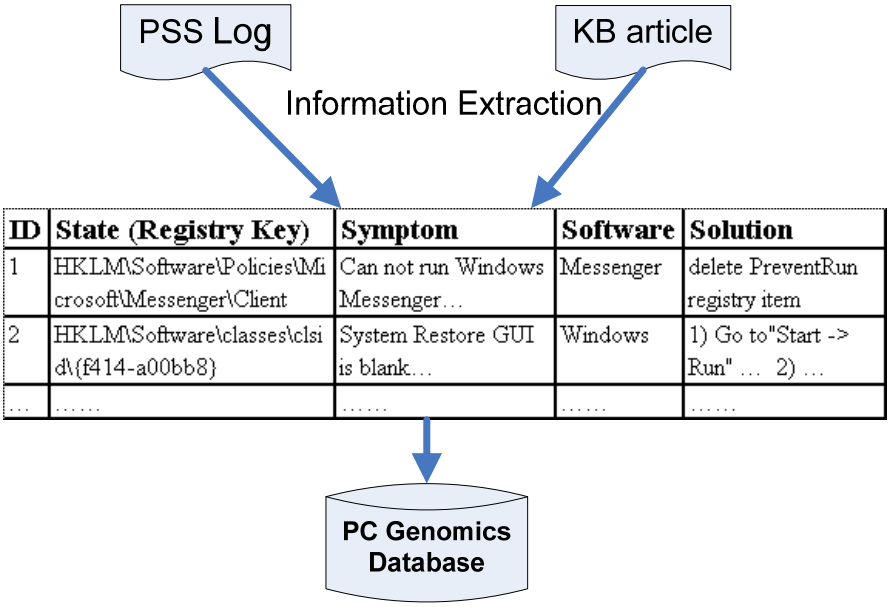

图 3.5 抽取计算机“基因库”

3.2.3 利用计算机基因库生成问题可能的原因的排名

这一部分将介绍对于一个新的用户报修问题，如何利用基因库中储存的信息来进行诊断。

基因库储存了一系列曾经发生过问题的静态信息 RC_1, RC_2，RC_3…。每个静态信息都有其相关的一系列症状描述 $\mathbf{S}_{i,1}$，$\mathbf{S}_{i,2}$，$\mathbf{S}_{i,3}$,…（图 3.6）。当一个新的问题发生时如果用户能够对其症状做文字上的描述 $\mathbf{S}_{current}$，则这个描述就可以同基因库中各个静态信息的描述相比较，按照文字的相似程度判定哪一个静态信息更有可能是问题的原因（如图 3.6）。

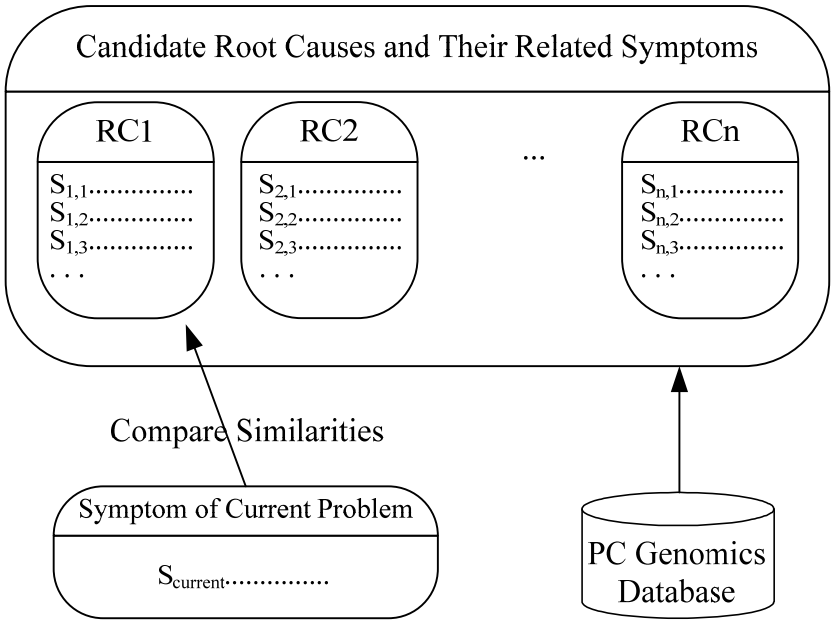

图 3.6 候选静态信息的过滤和评判

文本相似度的计算采用文本向量化后的内积（公式 3-1）。任意一段文字描述都被抽象为所有单词构成的空间中的整数向量。在基因库中同一个静态信息对应的历史症状文字都被加和成为单一的文本向量 $\mathbf{S}_i = \mathbf{S}_{i,1} + \mathbf{S}_{i,2} + ... + \mathbf{S}_{i,n}$。这个公式基于这样一个基本假设：如果一个静态信息是当前问题的原因，则很有可能这个静态信息在之前的历史记录中曾产生过相似的症状。

$$similarity(S_{current}, RC_x) = S_{current} \bullet \sum_i S_{x,i} \tag{3-1}$$

where $S_{current}$ and $S_{x,i}$ are the vector representation of text

在实际应用中问题真正的原因如果能够被排在前 5 名则能够很好的帮助工程师进行诊断。然而在 Strider 工具中参加排名的候选静态信息是简单按照其在跟踪集中出现的顺序排名的。

3.2.4 只使用比较集或跟踪集进行诊断

一般说来参加排名的候选静态信息来自用户端收集来的比较集。但是有的情况下由于无法读取系统还原点，或者问题无法重现等问题 Strider 只能采集到

跟踪集或者比较集。在这种情况下 Strider 工具候选静态信息集合的大小一般都在几千的量级，则工程师不可能从中分析真正的问题原因了。

然而如果利用算机基因库的知识作为有效的过滤则可以在收集信息不完全时使得系统仍然能够有效的帮助诊断。首先，如果跟踪集无法收集，那么可以将算机基因库的排名算法用于比较集。其次，如果比较集无法收集，那么可以将算机基因库的排名算法用于跟踪集。最后，如果比较集和跟踪集都能够秤钩收集，那么可以将算机基因库的排名算法用于交集。这个排名结果就能达到最优的诊断精度，满足工程师的需求。

3.2.5 试验结果以及分析

本文利用 74 个注册表相关的问题来测试我们的系统。每个问题都利用 Strider 工具收集了问题文字描述、检测集合、比较集合。算机基因库则分别用 PSS 记录或者“微软知识库”的数据建立。

图 3.7 为利用 PSS 记录时四种方法诊断效果曲线的比较。从左到右四条曲线依次是：跟踪集的排名效果，比较集的排名效果，Strider 工具利用比较集的排名效果，以及比较集的排名效果。图中每个点代表一个 74 个问题中的某一个。纵轴表示问题中真正的病因在候选集合中被系统排到了第几位。而横轴则是所有问题按照其病因排名的效果来排序。通过排名曲线我们可以容易的比较不同排名方法之间的优劣。74 个问题在 PSS 记录中有 59 个有历史记录，所以余下的 15 个则无法得到计算机基因库的帮助。

首先我们可以看到使用交集的情况下 Strider 系统只能将 59 个问题中的 35 个排到前 5 名。而利用计算机基因库计算的症状相似度可以帮助 59 个问题中的 50 个排到前 5 名。这将明显加快工程师解决问题的速度。另外，在只使用比较集或跟踪集进行诊断的情况下有超过 1/3 的问题可以取得较好的诊断效果。虽然相对 Strider 系统是有进步，却仍然有较大的提升空间。

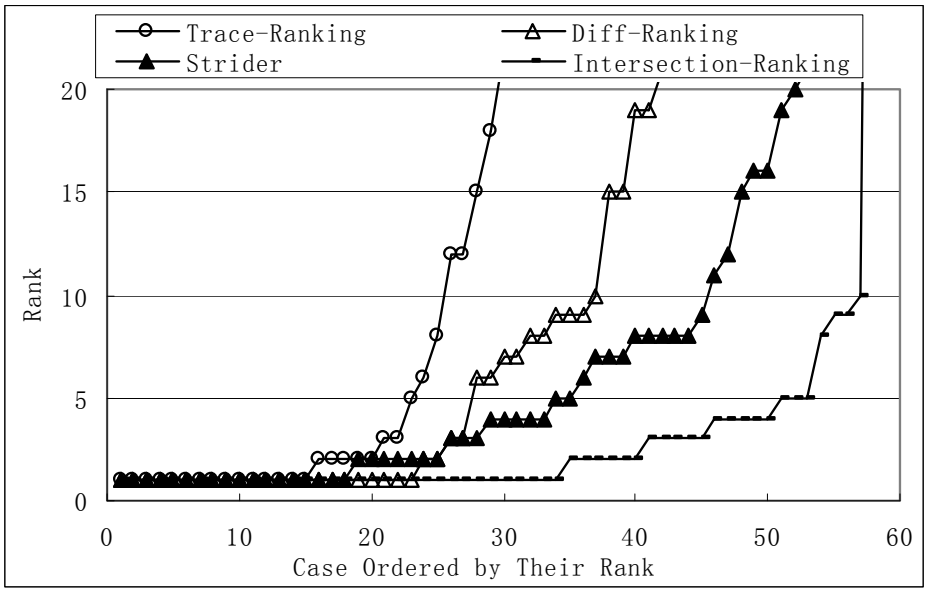

图 3.7　利用 PSS 记录的排名结果

图 3.8 为利用“微软知识库”时四种方法诊断效果曲线的比较。利用“微软知识库”建立的计算机基因库只能涵盖 74 个病因中的中的 36 个。可见虽然“微软知识库”的数据质量较好，覆盖面却不及 PSS 记录。

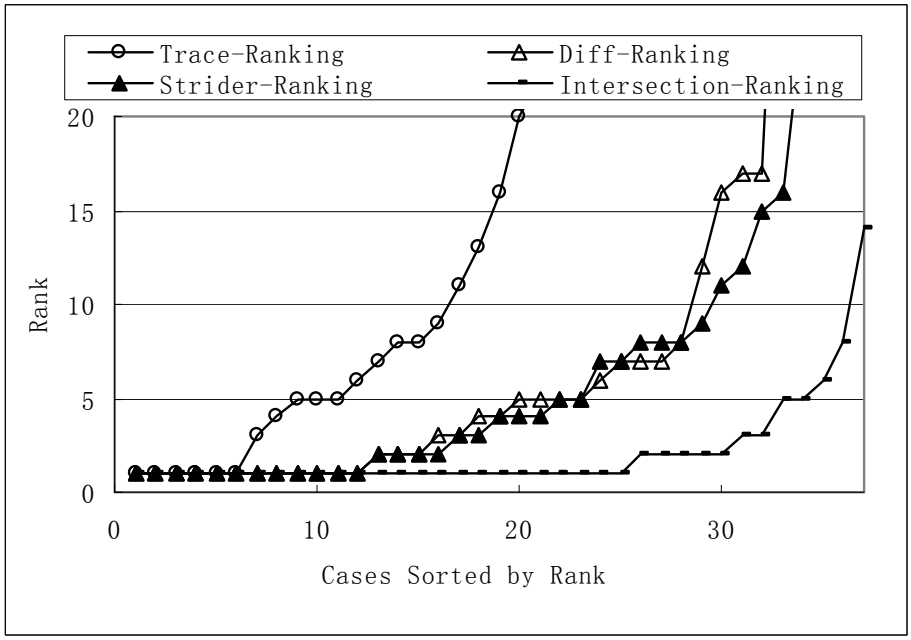

图 3.8　利用“微软知识库”的排名结果

总结起来，以上实验表明计算机基因库能够在使用交集的情况下明显的提高 Strider 系统的性能，而且在收集数据不完整的时候仍然能够帮助部分问题得到很好的解决。

3.3 在知识库中查找问题相关的文献

利用计算机基因库除了可以直接推测问题的原因，还可以帮助在知识库中查找问题相关的文献。这一功能可以帮助缺乏经验的工程师或者用户通过阅读问题相关文档了解问题。这一节将介绍如何将此问题构造成为“上下文查询”的问题。最后通过四个不同的概率模型比较试验显示出计算机基因库在诊断此问题上的良好性能。

3.3.1 上下文查询问题

上下文查询（Contextual retrieval）是信息检索领域所面临一个长期挑战[29]。Allan[29]将它定义为“将搜索技术与用户和查询的上下文结合一个整体的框架，并给予用户信息需求以最恰当的答案”（“combine search technologies and knowledge about query and user context into a single framework in order to provide the most ‘appropriate’ answer for a user's information needs”）. Pitkow 等[28]描绘了上下文查询能够给未来搜索技术带来的革命性变化。上下文查询在诸多领域都充当着关键性的技术——例如移动搜索(mobile search)、个性化搜索(personalized search)、计算机诊断(PC troubleshooting)等。然而，长久以来还没有工作给出上下文查询过程的完整模型。

Lawrence[47]讨论了互联网信息检索中上下文查询的一些普遍问题。Finkelstein [48]提出了从文档中选取文字作为查询，并且以周围文字作为上下文进行查询的办法。这里周围文字被作为查询的扩展，用扩展后的查询进行检索。

据我们所知，在之前的工作都没有能够完整的给出上下文查询中查询、上下文、以及需要检索的文档的关系。也没有将上下文查询应用于实际问题的例子。我们以计算机自动诊断为背景提出了四个上下文查询模型，并且成为首先成功的实现上下文查询并应用于实际问题的例子。

在上下文查询中有三个主要的元素：查询、上下文和需要检索的文档。对应到计算机诊断问题里则分别是用户对问题的描述、系统静态信息和知识库中

问题相关的文献。本文提出的四个上下文查询解决了困扰上下文查询的三个问题[15]：

1. *异质的上下文*: 在之前的上下文查询和个性化查询中，查询、上下文和文档都由同样的元素——词——构成。这样就很容易将上下文当作查询词的扩展嵌入到信息检索的框架中[30][31]。但是对于广义的上下文查询来说，上下文却不一定是文字。比如计算机的状态，手机用户的位置，当时的天气温度等等。所以之前上下文查询的方法的使用是有很强的限制的。

2. *无关的上下文*: 多数之前的工作假设所有上下文都是同查询相关的。这只在少数情况下是可行的。例如用户主动提供个人档案的时候。但是对于自动收集的大量上下文来说（在我们的例子中可能是上万的注册表项）多数上下文同当前查询是无关的。这些无关上下文则可以对检索带来负面效果。从另一个视角，该问题可以理解为区分上下文相关和无关的查询。若能够准确的过滤无关的上下文，那么那些上下文相关的查询可以从上下文中得到提高，并且那些上下文无关的查询也不会受到负面影响。

3. *不完整的查询*: 在信息检索中一个普遍的问题是用户很少输入完整的查询来描述他们的信息需求[32]。这主要是来自于用户的惰性和缺乏经验。对于移动搜索来说，用户是难以在有限的输入设备上完成这个工作。对于计算机诊断问题用户更是无法看到系统的静态信息。所以上下文查询可以弥补查询信息的不完整，提高信息检索的效果。

3.3.2 本文提出的四个上下文查询模型

我们通过概率模型来产生查询扩展词(query expansion term)。当一个新的查询以及当时的上下文被送到搜索引擎，一系列的文档词将被选出来并按照它们相关的条件概率排序。最相关的则被当作查询扩展词。

模型 1:上下文模型（Context Only Model）

这是一个符合直觉的简单模型：文档中和上下文密切相关的词被选作查询扩展词。

$$M_1(d \mid < Q, C >) = I(d, C) = \sum_{c_i \in C} I(d, c_i) \tag{3-2}$$

其中 *I(d, C)* 是文档中某个词同当前上下文的互信息。从查询同上下文结合的方式来看，这个模型很类似于传统的上下文查询方案。但是在我们的问题里扩展词不是来自于上下文（因为上下文并不是由语言描述的），而是来自上下文相关的文档中。这样我们就解决了上下文不是自然语言的问题。

模型 2: “查询－上下文”无关模型(Query-Context Independent Model)

模型 1 的缺点是扩展词完全来自于上下文并且同查询词无关。由于某个上下文可能同时出现在多个不同的查询中，同某个查询无关的词也就可能被选中。第二个模型则增加了对查询词相关性的考虑。

$$\begin{aligned} M_2(d \mid < Q, C >) &= I(d, < Q, C >) \\ &= I(d, C) + I(d, Q) \\ &= \sum_{c_i \in C} I(d, c_i) + \sum_{q_j \in Q} I(d, q_j) \end{aligned} \tag{3-3}$$

其中 *I(d, Q)* 是文档中某个词同当前查询的互信息。相对于模型 1 来说，这里的优势在于赋予对查询词相关的词以更高的权重。在公式中，由于数据稀疏性造成直接计算联合概率 *P(d, <Q, C>)* 比较困难。所以我们引入了三个独立性假设：查询词间的独立、上下文间的独立性、查询词间和上下文间的独立性。虽然前两条是比较合理的，而且常常被用于之前各种概率模型中，第三条却是不利的。例如文档中某个词可能偶然同一些上下文有较高的联系，则即使是在它同当前查询完全无关的情况下也会被选中。

模型 3: “查询－上下文”关联模型(Query-Context Dependent Model)

为了减弱查询词间和上下文间的独立性假设，我们提出模型 3。

$$\begin{aligned} M_3(d \mid < Q, C >) &= I(d, < Q, C >) \\ &\propto \sum_{c_i \in C} I(d, c_i) + \sum_{q_j \in Q} I(d, q_j) + \alpha \sum_{c_i \in C, q_j \in Q} I(d, < q_j, c_i >) \end{aligned} \tag{3-4}$$

其中 $\sum_{c_i \in C', q_j \in Q} I(d, < q_j, c_i >)$ 是文档中某个词同当前查询－上下文对之间的互信息。这一项就引入了查询词间和上下文间的关联性。由此那些与查询词间和上下文同时相关的文档词将得到额外的权重。参数 α 用于调整这种相关性的大小。

模型 4: 上下文协同过滤模型（Context Filtering Model）

模型 1、2、3 遇到的一个共同问题是无关上下文构成的噪声并没有得到处理。显然无关上下文很容易引入无关的扩展词。模型 4 则过滤掉所有同查询相关度较小的上下文。

$$\begin{aligned} M_4(d \mid < Q, C >) &= I(d, < Q, C >) \\ &\propto I(d, < Q, C' >) \\ &\propto \sum_{c_i \in C'} I(d, c_i) + \sum_{q_j \in Q} I(d, q_j) + \alpha \sum_{c_i \in C', q_j \in Q} I(d, < q_j, c_i >) \end{aligned} \tag{3-5}$$

where

$$C' = \{c \mid c \in C, I(c, Q) \geq \varepsilon\}$$

其中 *I(c, Q)* 是某个上下文同当前查询的互信息。当它小于一个阈值 ε，则该上下文会被滤掉。该模型的直觉解释是：如果某个上下文在以前的记录中很少与当前查询词相关则它就不再被用作查询扩展。

3.3.3 试验设置

文档集合来自于微软知识库中 142,448 篇工程师写的文章。PSS 记录中的 31,933 个注册表相关案例被用于训练模型。每个案例包涵了用户问题描述、相关注册表项、以及工程师在处理该案例后推荐的文档。所以每个案例都包涵了完整的查询－上下文－文档序列。

根据这些数据我们用最大似然法计算了如下的互信息：*I(d, q), I(d, c), I(q, c), I(d, <q, c>)* 。

$$I(d, q) = \ln \frac{f(d, q) \times M}{f(d) f(q)}$$

$$I(d, c) = \ln \frac{f(d, c) \times M}{f(d) f(c)} \tag{3-6}$$

$$I(q,c)=\ln\frac{f(q,c)\times N}{f(q)f(c)}$$

$$I(d,<q,c>)=\ln\frac{f(d,<q,c>)\times M}{f(d)f(<q,c>)}$$

其中 *f(q), f(c), f(d, q), f(d, c), f(q, c), f(d, <q, c>)* 是上下文、文档词、查询词等元素所出现过的案例数。*f(d)*是词 *d* 所出现过的文档次数。*M* 是总文档数。*N* 是总案例数。.

传统的统计学习中平滑（smoothing）技术通常被用于解决训练过程中数据稀疏的问题。在我们的问题中 31,933 个案例相对与参数的个数还是相对较小的。因为互信息倾向于给罕见的数值赋予较高的权重，所以我们滤掉了所有出现次数小于 10 的词和上下文。对于 *f(a, b)=0* ，我们定义 *I(a, b)=0*。

我们从网上收集了 29 个真实用户的案例用于测试（表 3.1）。这些问题不是来自于 PSS 工作记录所以同我们的训练集是独立的。每个案例都收集了完整的查询和上下文。相关的文档由工程师人工选择出来。

表 3.1 29 个测试用案例

No.	Query	Context*
1	Cannot open Outlook attachment	29
2	Blank Windows activation page	27
3	IE always launch in offline mode	26
4	Spell checker does not work in Outlook Express	13
5	Title bar text changed after visiting some websites	78
6	Using Word as the Outlook email editor, cannot turn off Word ‘Document Map’ option	26
7	Cannot open address book in Outlook	39
8	Unable to play file with media player	41
9	Cannot open .exe programs	24
10	Outlook cannot remember password	38
11	Windows Messenger stops running	26
12	'File not found' message potentially due to a virus or misconfiguration in antivirus program	14
13	Not launch browser when click Internet in Start menu	12
14	Office 2000 patch error	35
15	Error 1606 when installing and uninstalling	18
16	Excel cannot startup	17
17	Can not change IE home page settings	13
18	Office shortcut bar missed	13
19	MSN Explorer dial-up unable to sign in using analog phone line	21
20	Duplex Printer becomes simplex printer	16
21	IE can only save image file as bmp file	32
22	Desktop Tab missing from Display Properties	24
23	CD ROM error	15
24	Windows Media player product feedback causes AV	14
25	Windows Update access denied	29
26	‘Search’ prompts when double clicking a folder	24
27	Start System Restore, blank window appears	19
28	Cannot create a new dial up connection	26
29	Cannot connect to internet	51

*由于空间原因这里没有写出具体的注册表项，而代之以注册表项的个数。

排名算法我们采用 Okapi 的 BM2500 [33]作为信息检索系统。 基本公式如式 3-7：

$$\sum_{T \in Q} w^{(1)} \frac{(k_1 + 1)tf\,(k_3 + 1)qtf}{(K + tf)(k_3 + qtf)} \tag{3-7}$$

其中 Q 是包涵词 T 的查询， tf 是词在某文档中的出现次数，qtf 是查询中某词的权重，$w^{(1)}$ 是 T 在 Q 中的 Robertson/Spark Jones 系数. K 由式 3-8 给出。其中 dl 和 $avdl$ 是文档长度和平均文档长度。在所有试验中我们均取 k_1 = 1.5, k_3 = 5, b = 0.5, $avdl$ = 200。

$$k_1((1 - b) + b \times dl / avdl) \tag{3-8}$$

作为对照，传统未采用上下文检索的结果用作对照组。四种模型分别选出 10 个查询扩展词。原本查询中的词取 qtf＝5，扩展词统一取 qtf＝1。模型 3、4 中 α＝0.5。模型 4 中 ε=1.0。试验中采用 Porter stemming 并且滤掉 426 个禁止词（stop word）。只考虑了单词而不是词组。

这里比较特殊的是每个案例在知识库中只有一篇对应的文档。所以我们人工为每个案例标出了知识库中唯一包涵了解决方案的文档。然后我们计算了正确答案能够被系统排名到 1、5、10、20、50、100 的案例比例。特别的，如果答案被排到 10 名之外，那么用户将很难找到它。

3.3.4 试验结果以及分析

表 3.3 和 3.4 集中列出了 29 个案例在不同模型下的检索结果试验结果。

首先，我们可以看到对照组的排名结果是最差的。只有 41%的案例能排到结果集合的前 10 名，而且没有一个被排到第一。可见在这个应用上采用传统的信息检索方法而不使用上下文信息是很难取得满意的效果的。因为这里的查询信息大多多是不完整的。例如 Q9 “Cannot open .exe programs” 是对问题非常笼统的描述。相反对于 3, 5, 11, 15, 20, 22 和 25 这些排名进入前 5 的案例来说，他们的查询通常提到了可能的问题原因。对于这些比较明确的查询，上下文的作用就不大了。

表 3.2　29 个案例在不同模型下的检索结果

Model	Base line	Model 1	Model 2	Model 3	Model 4
Q1	132	20	20	5	1
Q2	7	1	1	1	1
Q3	5	10	18	12	6
Q4	16	8	3	6	1
Q5	3	5	6	6	3
Q6	29	8	7	8	6
Q7	153	191	200	55	1
Q8	150	2	2	2	4
Q9	4,697	2,039	1,308	372	1
Q10	51	4	3	3	3
Q11	5	8	5	2	6
Q12	30	2	2	2	2
Q13	11	12	22	16	3
Q14	32	5	5	7	2
Q15	4	1	1	1	1
Q16	11	13	2	3	1
Q17	9	1	1	1	1
Q18	31	4	3	1	1
Q19	9	5	4	4	4
Q20	5	7	6	3	6
Q21	17	21	18	13	18
Q22	3	3	3	3	3
Q23	1,024	15	5	4	9
Q24	7	8	21	12	21
Q25	4	12	32	36	26
Q26	22	1	1	1	1
Q27	7	3	1	1	1
Q28	951	7	7	5	1
Q29	1,410	8	3	4	1

第二，仅仅通过引入上下文信息，模型 1 取得了较大得提高。可见上下对于寻找问题相关的查询扩展词是很有效的。但是由于同时无关上下文的引入，一些案例的效果更差了。例如 Q3、Q5、Q11 等。

第三，模型 2 同时使用了上下文和查询的信息，所以得到了比模型 1 更好的效果。在进一步引入了上下文和查询的交叉项信息后，模型 3 又取得了进一步的提高。

最后模型 4 显示出滤出无关上下文的重要作用。大约 45%的案例能将答案排到第一，90%的能排到前 10。这样的精确度能极大的提高售后服务工程师在诊断中的效率。这一成功部分是由于这一应用中的静态信息中存在的大量无关信息。在其它领域，特别是象移动搜索等自动收集上下文的应用中，无关信息的问题依旧是很严重的。所以查询相关的上下文过滤仍旧会成为重要的技术。

表 3.3 被排在前 1、5、10、20、50、100 的案例个数

Model	**Top 1**	**Top 5**	**Top 10**	**Top 20**	**Top 50**	**Top 100**
Baseline	0	0.24	0.41	0.55	0.72	0.76
Model 1	0.14	0.45	0.69	0.9	0.93	0.93
Model 2	0.17	0.59	0.72	0.83	0.93	0.93
Model 3	0.21	0.62	0.76	0.9	0.93	0.97
Model 4	0.45	0.72	0.9	0.93	1	1

3.4 讨论

这一节中我将讨论计算机基因库之所以有效的两方面原因：第一是基因库的筛选作用；第二是基因库对于所有问题的有效覆盖率。

3.4.1 静态信息如何帮助筛选病因

仅仅依靠问题症状描述进行诊断的的缺点在于存在很多问题有同样的症状却有不同的病因。例如在售后服务记录中我发现有 17 种原因对应着同样的症状“Cannot open Word document” (表 3.4). 如果单依靠问题症状描述进行诊断，则我们将得到多个同样可能的病因。这时系统静态信息就能充当有效的过滤系统，指出正确的病因。

表 3.4 “Cannot open Word document”问题的病因

No.	Root Cause
1	hkey_users\.default\Software\Microsoft\Office\8.0\Outlook\Options\Mail
2	hkey_current_user\software\microsoft\office\9.0\word\data\toolbars
3	hkey_current_user\software\microsoft\office\9.0\word\data\settings
4	hkey_current_user\software\microsoft\office\10.0\word\data\settings
5	hkey_current_user\software\microsoft\office\10.0\word\data\toolbars
6	hkey_local_machine\SOFTWARE\Microsoft\Internet Explorer\Plugins
7	hkey_current_user\environment
8	hkey_local_machine\System\CurrentControlSet\Services\Inetinfo\ Parameters\MIMEMap
9	hkey_classes_root\.doc\Content Type
10	key_classes_root\mime\database\content type
11	hkey_classes_root\MIME\DATABASE\Charset
12	hkey_classes_root\MIME\DATABASE\Codepage
13	hkey_classes_root\word.document
14	hkey_classes_root\excel.sheet.8\shell\open\command
15	hkey_current_user\software\microsoft\office\9.0\common\general\startup
16	hkey_local_machine\software\microsoft\shared tools\text converters\import
17	hkey_local_machine\software\microsoft\shared tools\text converters\ import\wordperfect6x
18	hkey_classes_root\excel.sheet.8\shell\open\ddeexec
19	hkey_current_user\software\microsoft\shared tools\outlook\journaling\ microsoft word\autojournaled

另外，在售后服务记录 PSS 中我发现只有大约 0.5%的注册表项被报告发生过问题 (898,546 中的 4,837)。在知识库 KB 数据中只有 0.1% (898,546 中的 996)。除去大量“记录良好”的系统信息也是基因库能构有效缩小可能病因集合的原

因。

3.4.2 有限的基因库如何能帮节省大部分人力资源

基因库的有效性取决于其涵盖用户真实遇到问题的比例。在 4,837 个在 PSS 数据里被记录的注册表项中我们发现只有少数频繁的发生问题，而多数都只有很少的次数（图 3.8）。 它们大致的服从 Zipf 分布[27]，即某种问题发生的概率同其排名在双对数图上成斜率为-1 的线性关系。PSS 数据中共含有大约 10^7 问题记录。由此可以计算，如果数据库涵盖了前 1%种最普遍的问题则可以帮助解决 67%的售后服务问题。所以即使是较小型的基因库，通过实现自动诊断也能极大的节约售后服务成本。

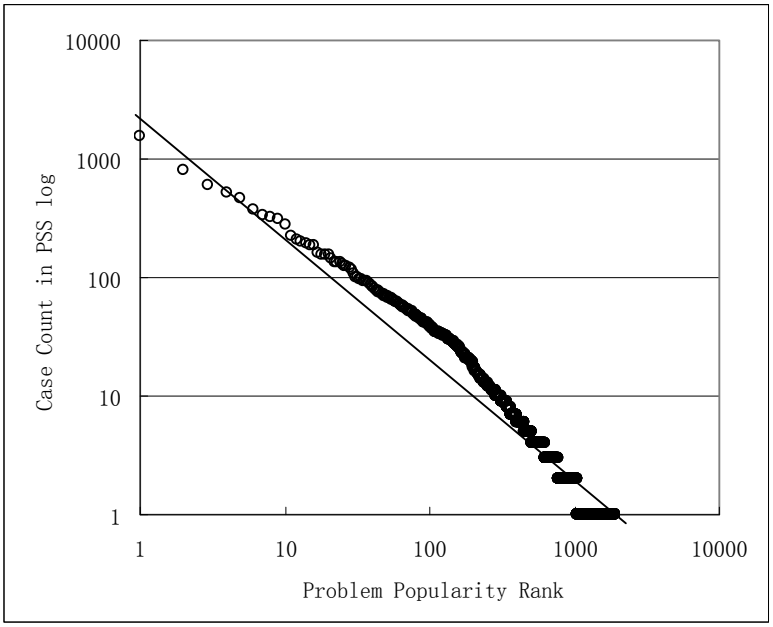

图 3.8 注册表项在 PSS 数据中出现次数分布

第4章 基于动态事件的诊断

本章将介绍如何收集并利用系统动态事件来进一步提高诊断的精确度和问题覆盖面。基本的思想是结合错误症状的检测和系统调用序列的分类以实现完全无需人工干预的自动诊断。这个方法应用在 Windows 系统的 5 个常见的问题上都取得了良好的效果。最后我分析了该方法可以进一步改进的几个方向。

4.1 基本原理

*动态事件*是指计算机在运行过程中产生的可以观测的现象。包括进程、线程的切换，用户函数、系统函数的调用，内存、文件、CPU、网络等资源的分配等等。

依据系统的行为而非系统的静态信息来诊断计算机问题就好像依据病人的疾病特征来诊断病因。比如要判断一个人是否感冒，不需要在显微镜下看见感冒病菌而只需要根据其体温，心跳，面色等外部行为就可以判断了。

基于动态事件诊断具有比用户文字描述更丰富的信息，而且相对于静态信息有以下的优势：

1. 更少的无关信息。静态信息的收集是通过扫描系统的所有状态然后依靠各种手段滤出无关的部分。由于所有无关的信息都参与到了这个过程中，引入噪声就在所难免。然而动态事件一般是在系统问题发生的同时收集信息，只有少数涉及到问题的信息被收集了。
2. 大大降低用于诊断所需要收集的信息量。静态信息的收集需要扫描系统的所有状态，这些信息的数量是巨大的。比如一般个人电脑的注册表项约为 107 个。动态事件的数量级在 103～104 左右，大大简化了计算量。
3. 解决许多静态信息无法收集到的问题。在许多情况下，信息收集工具无法收集造成问题的系统信息。比如有网络的环境下，问题可能存在于服务器端，或者网路交换设备上。再比如问题出在格式无法识别的第三方软件配置文件里面。如果在 Windows 这样有集中软件设置服务的操作系统上这个问题还不是非常严重，则在 Linux 等软件设置分散存放的操作系统上这个问题则是致命的。

4. 可以实现完全无需人工干预的自动诊断。根据上一章的结论，基于静态信息诊断的准确性依赖于用户对问题症状的描述。这就意味着诊断过程需要人工的干预。问题症状主要分为用户的行为和系统的响应。它们都是可以通过操作系统的技术自动检测的。由此来替代用户的输入则可以实现完全无需人工干预的自动诊断

但是，由于分析的复杂性，动态事件还没有被应用于个人电脑上的诊断中。现有的工作集中在计算机群和网路等较复杂的系统中利用关联性分析、聚类、分类等一系列数据挖掘方法，帮助问题分析，却不能实现自动诊断。本文提出两层分类器的结构以实现完全无需人工干预的已知问题自动诊断。第一层分类通过错误症状的自动检测实现。第二层通过系统调用序列的分类来实现。这两个部分将在本章中节分别介绍。

4.2 利用错误症状检测推测问题的原因

4.2.1 错误症状的自动检测

错误症状的自动检测通过Windows钩子技术[11]和DLL注射技术我们可以实现系统中用户的行为和系统的响应的捕捉。

钩子技术在第 2 章中已经介绍。它可以在系统核心态截获所有系统调用相关的信息。然而，在 Windows 系统的设计中用户界面部分是 shell 来完成的，属于用户态而非核心态。所以多数用户界面相关的事件，例如用户键盘鼠标的输入、产生窗口和菜单是无法通过钩子技术截获的。

DLL 注射技术却能很好的解决这一问题。其基本原理是通过 Windows API 将一个具有检测功能的 DLL 加载到所有用户太的进程中。这些进程收发的所有消息都将被该 DLL 截获。这些信息经过分析后可以从各个进程发回到收集数据的进程并记录下来。

据我们所知，本文是第一个利用 DLL 注射技术帮助实现自动诊断的工作。

以下列举了几种常见的问题症状信息：

1. 新进程的版本和文件

Explorer.EXE　SHLWAPI.dll　6.00.2900.2180 (xpsp_sp2_rtm.040803-2158)

Explorer.EXE　SHLWAPI.dll　C:\WINDOWS\system32\SHLWAPI.dll

- 新窗口的产生和显示

msimn.exe　msoe.dll　　wm_create outlook express

msimn.exe　msoe.dll　　wnd:outlook express

- 用户点击按钮或菜单:

iexplore.exe mshtml.dll　Item:Find

2. 进程的产生和消失

Explorer.EXE　　SHLWAPI.dll　　CREATE_PROCESS

Explorer.EXE　　SHLWAPI.dll　　END_PROCESS

利用这些事件我们可以很准确的识别某个问题中的症状。图 9 给出了从系统事件识别问题症状的过程。这里我们定义每个症状是一个或多个事件的与。于是症状的识别即可通过其所对应的事件是否都发生了来判断。例如“在 word 中不能打印”这个症状可以表达为“用户在 word 进程中启动打印对话框”AND “word 进程弹出无法打印的对话框”。这样通过识别问题症状我们则可以基本判断问题的原因，并得出解决方法。

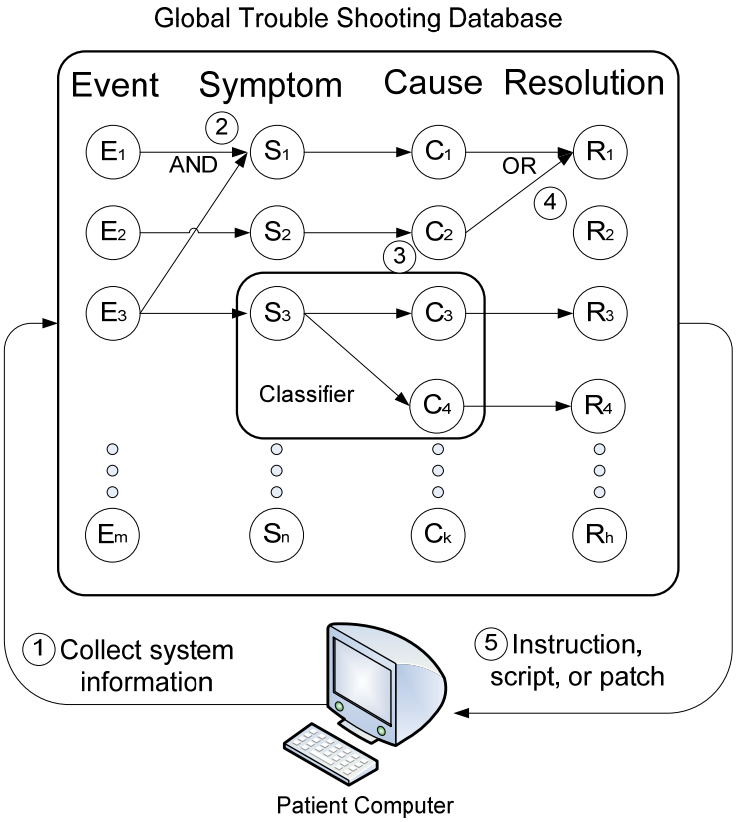

图 4.1　症状识别系统

这里的问题是同一症状有可能会对应不止一个原因。例如 1.2 节 IE 问题中那样。根据从微软“知识库”中统计的结果约有 10%的问题有超过一个原因。这时就需要收集更多的系统信息构建分类器来区分它们。如何构建分类器将在下一节中讨论。

4.2.2 试验设置以及结果

为了验证症状识别系统的性能我们从微软问题反馈数据库中选取了 102 种一般用户比较常见的问题症状。这些问题涵盖了 10 类软件（图 4.2，表 4.1）。由于空间原因这里只列出了每种软件间相关的症状数而不是具体症状。

图 4.2　问题反馈数据库

表 4.1　102 种常用的用户操作

Product	**#user actions**
Internet Explorer	9
Outlook Express	7
Accessory Programs	17
Access	12
Excel	11
FrontPage	1
Outlook	15
PowerPoint	13
Visio	7
Word	16

采集数据的方法是人工重现每一个问题症状同时打开数据采集程序。并且重复两次以上操作。所以共得到 204 条测试数据。每个症状所对应的系统事件由人工选取。例如在“用户在 Word 中无法打印”这个问题中，我们选取“用户鼠标点击名称为 print 的按钮”以及“系统弹出带有 Cannot print 字样的对话框”为需要识别的系统事件。

试验结果表明系统在这些症状的识别结果可以达到 precision=95%，recall=100%。即所有症状都得到了识别，并且只有极少数的情况下顺带错误的

检测到了其它不相关的症状。可见该方法的准确度和效率都是很高的。对于这种误判的情况，我们还可以依赖第二层分类器进一步祛除不相关的可能性。

另外该症状识别系统对于每个症状需要收集的事件数目大都小于 103，远远小于收集静态信息的数量。这也体现了动态信息中含有更少无关信息的优点。

4.3 利用系统调用序列推测问题的原因

由 1.2 节中 IE 的例子我们会发现，一些问题拥有相同的症状却对应着完全不同的原因。这就提示我们单单利用症状的检测无法准确诊断所有的问题的。我们需要收集更多的系统信息构建分类器来区分它们。

为了分析计算机自动诊断中的系统动态事件，本文系统的研究了序列数据分类问题中的几种特征表示方式的有效性。特别的，本文提出了一种新的基于字符串比对的序列数据特征抽取方法。结合支持向量机分类器，该特征抽取方法明显优于传统的 n-gram 序列特征抽取方法。同时也优于隐马尔可夫模型等经典序列分类方法。

在 Windows 系统的 5 个常见的问题上的实验结果也有力的表明，顺序关系在序列数据中起着非常重要的作用，而且字符串比对是一种能发现长距离相关性并且祛除序列中广泛存在的噪声的有效方法。

4.3.1 系统调用序列

我们知道操作系统负责计算机的信息交互、资源分配等基本功能。一个应用程序如果要实现进程间消息通讯、读写 IO、读写磁盘、申请或释放内存等功能都要通过调用系统函数。通过钩子技术， 我们就能记录所有程序对系统函数的调用。这些调用按照时间排列则成为含有丰富语义(semantics)的时间序列数据（表 4.2）。

表 4.2　系统函数调用序列

序号	进程	函数	参数	返回值	数据
6914	OUTLOOK: 3292	CloseKey	HKLM\... \Parameters	SUCC	0xE13B83B0
6915	AcroRd32: 2728	PeekMessage_20	WM_TIMER	-	Data
6916	AcroRd32: 2728	GetMessage_16	WM_TIMER	-	Data
6917	OUTLOOK: 3776	CreateKey	HKLM\... \Parameters	SUCC	0xE13C5450

一般来说，程序在运行过程需要大量的产生系统调用。例如“打开 Word 文档”一般包含了大约 3 万条系统调用，在 Internet Explorer 中浏览一张网页大约产生 9 万条系统调用。而且由于我们将一个系统调用的进程、函数、参数、返回值共同作为一个符号，所以该序列数据的字符集是很大的－对于每一问题一般都在几千的量级。

因为我们将 Strider 工具记录的系统调用序列按照进程、线程、时间的顺序排列，不同进程、线程的系统调用在之后的各算法中可看成是相互独立的。在文章余下的部分我们只讨论单个线程产生的序列。

Eskin 等[62]提出了“系统调用图”的模型来分析系统调用序列。在这个模型中，每个程序都有自身固定的“调用图”（图 4.3）。图中每条边都代表着一段程序代码。而结点则带代表了程序运行时可能分岔的地方。程序运行的过程就对应着调用图上从“开始”到“结束”的一条完整路径，叫做运行路径(execution path)。所以“调用图”集中代表了某个程序所有可能的运行路径。

由以上可以推测如果某个程序由不同病因造成的问题应该对应了不同的运行路径，而由相同病因造成的问题应该对应了相同的运行路径。而如果“调用图”中只有相关的系统调用被记录下来则形成了“系统调用图”。由于在“调用图”中每条边上的代码引起的系统调用在每次运行时都是固定的，分析系统调用则可能成为有效的分析运行路径的方法。

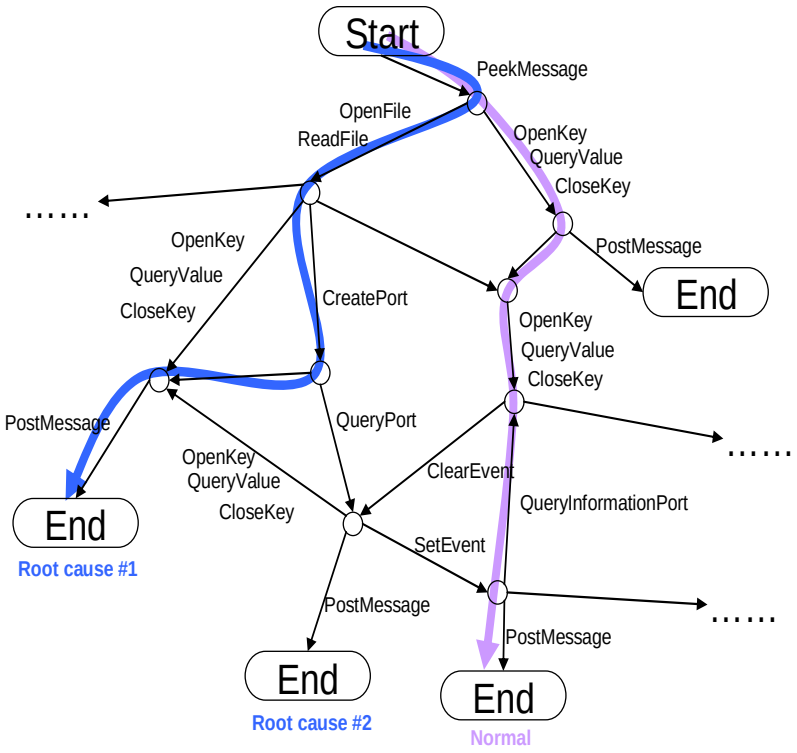

图 4.3　系统调用图

这里需要解决的问题是运行路径中的程序片断是连续运行的，而且调用的多是非常相似的系统函数。所以如何将其还原为片断并不是一件很容易的事。

4.3.2　序列数据分类问题简介

序列数据是在自然语言处理、生物信息学、入侵检测等领域广泛研究的数据结构。相对于其它类型的数据，顺序相关性在序列数据中起着更为重要的作用。并且序列数据往往具有高维的特点。所以在序列数据上的数据挖掘问题中合适的特征表示则显得尤其重要。既要能够保留元素间重要的顺序相关性，同时也要能有效率的处理高维数据的问题。

生物信息学以及入侵检测问题中的序列数据同自动诊断问题中的系统动态信息有着较多的相似之处，处理方法也最为值得借鉴。

4.3.2.1　生物信息学中的序列数据分类问题

在过去十年中，呈指数级增长的在线生物信息学数据以及其促进新药物研究的巨大潜力大大的促进了对序列数据分析问题的研究。为了实现海量序列数

据的标注(annotation)、分类、以及数据库检索，大量已有的数据挖掘方法以及机器学习方法被应用于生物信息学数据。同时各种新的方法也不断从生物信息学领域产生。特别的是，序列数据分类算法被广泛的应用于判断 DNA 编码区(coding regions)和非编码区(non-coding regions)、分辩外显子(exons)和内含子(introns)、预测蛋白质二级结构(secondary structure)、或是判断一个蛋白质序列所属的蛋白质家族等等[10]。

在这些应用中的一个共同的问题就是如何能够有效的从 DNA 序列、核酸序列、蛋白质序列等序列数据中抽取特征。这不是一个简单的问题。一方面，在缺乏空间向量表示的序列数据上经典的分类方法例如支持向量机和决策树等无法被直接应用。另一方面，将序列数据转化为空间向量表示的特征抽取过程却往往伴随着顺序相关性信息的丢失。同时也有一些专门为序列数据分析设计的算法，例如隐马尔可夫模型，则无需将序列数据转化为空间向量表示。

4.3.2.2　入侵检测中的序列数据分类问题

随着互联网的迅速发展，人们享受着访问大量数据的便利。然而，这种互连性同时给予有恶意的人利用软件安全漏洞侵入其他人电脑系统的机会。入侵检测即为通过监测、分析计算机系统中发生的事件来检测安全问题的方法[1]。入侵检测一般被抽象为一个分类问题：对于一条网络监听(network audit)序列或者系统事件(system event)序列，入侵检测系统(Intrusion Detection System)将其判别为正常或入侵两种行为。

入侵检测的概念最早由 Forrest 等[4]提出。他们采用了 n-gram (在信号处理领域即滑动窗口)的方法从序列数据中抽取特征。随后各种数据挖掘的方法也被相继应用到 计算机入侵检测中。其中大量使用了系统调用序列和符号序列分析方法，包括关联规则抽取 [5][6]，频繁自序列的挖掘[6]，隐马氏模型 HMM [6]，K 近邻分析 [7]，分段分析和聚类分析[8]等。其中很多都可以借鉴到自动诊断问题上来。

4.3.3　几种序列数据分类方法简介

我实现并比较了序列数据的词频特征表示，n-gram 特征表示，隐马尔可夫模型，以及序列比对分析四种方法。其中序列比对分析（Sting Alignment）近来被大量的用于生物信息学上，将在下一节介绍。本节主要介绍前三种方法。同

时介绍结合前两种方法使用的支持向量机分类器。

4.3.2.1　序列数据的词频特征表示

词频特征表示是最为简单的序列数据特征表示。该方法将序列数据表示为其包含元素的词频——词频向量。词频向量中的每一维表示了某一元素在序列中出现的次数。

这种方法在文本分类中得到很好的应用[55]。这种方法虽然丢弃了所有的顺序相关性信息，同支持向量机分类器相结合后往往能得到很好的分类效果。

4.3.2.2　序列数据的 n-gram 特征表示

N-gram 模型最早是在自然语言处理中得到应用[56]。直至今日，仍然广泛的被应用于信息检索领域中。其方法简单，并且能有效的从序列数据中抽取一定的顺序相关性信息[57]。

一个 n-gram 被定义为序列中任意连续 n 个元素。特别的当 n=1 时，该方法即退化为序列数据的词频特征表示。当 n=1,2,3,4 时，某个 n-gram 可以分别被叫做 bi-gram, tri-gram, 和 quad-gram。在信号处理领域 n-gram 方法一般被称为“滑动窗口”。

N-gram 模型遇到的主要问题时在尽可能保留顺序相关性和特征空间的维数之间的平衡。一方面取较小的 n 则会丢失较多的顺序相关性。另一方面，特征空间的维数同 n 呈指数级关系。过高的维数对于系统内存以及分类算法的实现都是很大的挑战。对于多数中等大小的问题或者采用了较好特征选取的大型问题，n 一般都不会超过 10。否则特征的储存对于当今的电脑系统都是无法接受的。

4.3.2.3　隐马尔可夫模型简介

隐马尔可夫模型是处理序列数据的一种有力工具。它拥有很好的统计学基础，但是需要对模型的拓扑结构进行一定的假设。在语言相关的研究领域中，HMM 得到了相当广泛的应用。例如语音识别[58]、名称识别[59]， 光学字符识别(Optical Character Recognition)[60]、以及信息抽取(Information Extraction) [61]等等。

一个隐马尔可夫模型主要包含以下的几个概念：

$\{S_1, S_2 \ldots S_M\}$表示模型所定义的系统可以存在的 M 个状态

$\vec{Y} = \{Y_1, Y_2, \cdots Y_T\}$表示一个可以观测的序列。

$\vec{X} = \{X_1, X_2, \cdots X_T\}$表示 Y 序列所对应的隐序列。每个时刻下 Xt 都是 S1, S2… SM M 个状态上的一个分布。

$\vec{\mu}$ 表示系统的初始状态分布，即 X_0

$A = \{a_{ij}\}_{M \times M}, a_{ij} = P(X_t = j \mid X_{t-1} = i)$为从任意一时刻 t 到下一时刻 t+1 时 X 的状态转移概率矩阵。

$B_{t,i} = P(Y_t \mid X_t = i)$为表达矩阵，即隐状态为 X_t 的条件下得到各个观测 Y_j 的概率分布.

$\vec{\lambda} = \{\vec{\mu}, A, B\}$即为构成一个完整隐马尔可夫模型的参数集合。

对应到计算机自动诊断的问题中，可以观测的时间 Yt 即为系统调用，隐状态即为的程序片断，而每种不同的病因对应了不同的运行路径即拥有为不同的隐马尔可夫模型 λ。我们采用了比较通用的“由左向右”模型(left-to-right model[63])（见图 4.4）。这种模型经过了较深入的研究，并且在我们的应用中具有较好的收敛性。

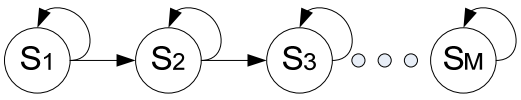

图 4.4　“由左向右”模型

利用 Baum-Welch EM [64]方法我们可以从训练数据集合为每种病因建立各自的隐马尔可夫模型（公式 4-1）。

$$\hat{\vec{\lambda}} = \underset{\vec{\lambda}}{argmax} \ \ln P(\vec{Y} \mid \vec{\lambda}) \tag{4-1}$$

之后对于任意需要被分类的序列，其最佳预测病因即为拥有最大似然概率的一类。似然概率通过 Forward-Backward 过程[64]来计算（公式 4-1）。

$$P(\vec{Y} \mid \vec{\lambda}) = \sum_{\vec{X}} P(\vec{Y}, \vec{X} \mid \vec{\lambda}) \tag{4-2}$$

4.3.2.4　支持向量机简介

支持向量机[53] 是 AT&T Bell 实验室的 Vapnik[53]提出的针对分类和回归问题的统计学习理论。它基于结构风险最小化原理(Structural Risk Minimization principle)，使得模型对期望数据做出最佳的界定，达到最佳的分类效果。由于许多引人注目的特点和出众的实验性能[54]，支持向量机方法越来越受重视。它具有适用于高维小样本问题的独特性质，近几年来被非常广泛的应用于科学技术的各个领域。而高维小样本正是系统诊断问题中数据的主要特征之一。所以我们以支持向量机为分类器结合不同的特征提取方法比较它们分类性能的优劣。

支持向量机模型使用 w 为法向量的超平面分开正负两类样本。决策函数为 f(x)= wT x+ w0。其训练过程的核心是求解如下带约束的优化问题:

$$\begin{aligned} &\min_{\vec{w}} M(\vec{w}) = \frac{1}{2}\|\vec{w}\|^2 + \gamma\sum_{i=1}^{N} e_i \\ &\text{约束条件 } e_i \geq 0, \quad y_i(\vec{w}^T\vec{x}_i + w_0) \geq 1 - e_i \quad i = 1,\cdots,N \end{aligned} \tag{4-3}$$

其中 y_i=+1 若 x_i 为正例，y_i=-1 若 x_i 为反例。e_i 是样本被错分的惩罚(图 1)。公式(1)中第一项$\|\vec{w}\|$代表正负样本在特征空间中被分开距离的倒数(图 1)，体现了模型的结构风险。第二项$\sum e_i$ 代表训练样本被模型错分的惩罚，体现了模型的经验风险。整个式子代表了支持向量机算法在结构和经验风险间寻求最佳平衡的过程。数值优化的理论可以证明该优化问题达到最优解时必然有一部分样本 xi 满足 $e_i = 0, \quad y_i(\vec{w}^T\vec{x}_i + w_0) = 1 - e_i$ 。这些样本即被称为支持向量(support vectors)。正是这些支持向量决定了分界面 w(图 4.5)。

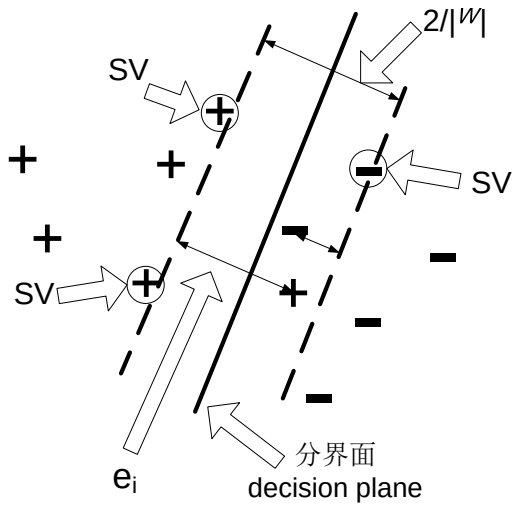

图 4.5 支持向量机(Support Vector Machine)原理图

e_i：错分样本惩罚；SV：支持向量；2/|W|：正负样本被分开的间距

4.3.4 序列比对分析抽取特征

序列比对分析抽取特征的方法首先使用编辑距离算法(edit distance)将已经标号的训练数据一一比对，并且合并成为一个标准序列（base sequence）（表 4.3）。然后将各训练数据按照同标准序列的比对转化成为统一维数的零一向量。利用模式识别中常用的分类器，例如支持向量机，则可以利用这些向量训练分类器。

表 4.3　基于比对的特征抽取（每个字母表示一个事件）

	Original	**Aligned**	**Feature Vector**
Root Cause 1	ABC	A B C	11100
Root Cause 2	ACD	A C D	10110
Root Cause 3	BDE	B D E	01011
Base Sequence		A B C D E	
Test Sequence	ABCC	A B C	11100

对于有待分类的测试序列，先将其与标准向量比对（表 4.3），转化成为零一向量。然后再将该零一向量送入分类器则可以得出其所属的类别。

图 4.6 显示了 90 条有关 IE 问题的系统调用序列被转化成为零一向量后的结果。从上至下共有 9 种不同的问题，每个 10 条数据。从左到右黑点代表 1，白点代表 0。可以很明显的看出在比对后的空间中不同类别问题带有各自不同的特征。

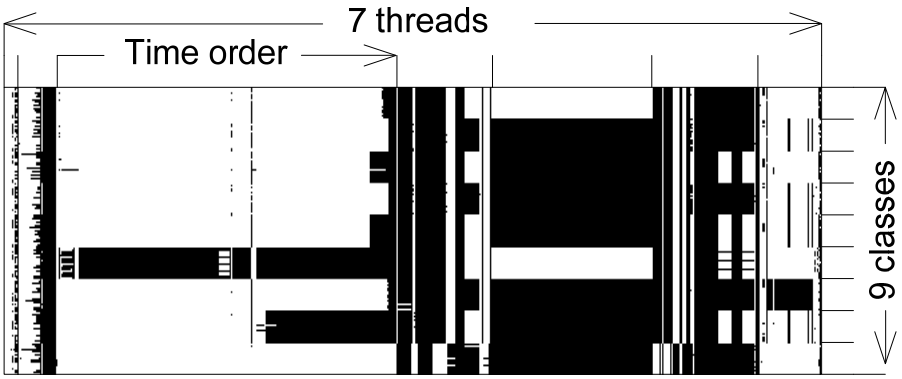

图 4.6 IE 问题的零一向量

在支持向量机训练之前，我们采用了噪声过滤：在标准向量的所有维中，那些在任意一类病因中出现都不超过 a%的维将被视为噪声而被剔除（a 为可以调控的参数）。在词频特征表示以及 n-gram 特征表示中我们也采用了相似的过滤方法：所有词中，那些在任意一类病因中出现都不超过 a%的将被视为噪声而被剔除。

这里标准向量所处的零一空间我们称为比对后的空间，而 n-gram 所使用的所有词构成的空间成为词频空间。比对后的空间上的噪声过滤由于考虑了事件的位置信息，所以效果要优于词频空间上的噪声过滤。

4.3.5 试验结果及分析

试验中我们重复试验了 5 类症状相同的问题如表 4.4。根据我们的经验，只使用注册表相关的系统函数将获得很有效的分类特征而且大大降低数据维数。比如“Page Cannot Display” in IE 这类问题有相同的症状，却有 9 个不同的病因。我们将每种病因产生的机理都研究透彻，然后在 17 台电脑上重现问题并且收集出问题时的系统调用序列。每台电脑，每个病因都收集 10 次，得到 10 条数据。

试验中我们发现对于同一病因但是来自不同电脑的数据会有相当的差异，

而来自同一台电脑的差异则较小。这是由于同一软件在运行过程中受到配置、版本或其他软件、插件的影响，运行路径可能会有不同。为了使得收集的数据不至于过于单一，我们试验的电脑不但包含实验室电脑，同时也包含同学宿舍和员工家庭电脑。

表 4.4　五个试验问题，类别数，试验机器数，
序列平均长度，注册表相关函数序列平均长度

Symptom	Class	Machine	Trace Length	
			All	Reg
Cannot Open Outlook Express	6	16	51,048	6,439
Cannot Share Folder	4	12	25,847	693
Cannot Print a Document	6	10	118,121	2,676
“Page Cannot Display” in IE	9	17	87,777	2,232
Error Opening Word	5	13	32,808	4,371

表 4.5 给出了不同方法分类的正确率。”*”号表示某类方法中效果最好的结果。Alignment 方法没有过虑噪声，“Alignment (A)” 以及 “Alignment (B)”分别代表了在词频空间上的噪声过滤以及比对后的空间上的噪声过滤。

首先，看 n-gram 方法。我们可以发现 n 大于一时总能比词频特征表示取得更好的效果。这是因为更多的顺序关系得到了保留。但是当 n 大于 3 以后分类效果往往反而下降。这表明虽然较长的顺序关系得到了保留，较短的顺序关系却丢失了。

第二，利用字符串比对的方法总是能够比 n-gram 取得更好的效果－正确率有平均 10%的提高。这是因为 n-gram 只能保留长度约为 n 的顺序关系，而字符串比对的方法却能保留任意长度的顺序关系。

第三，我们发现隐马尔可夫方法得到的分类效果普遍比较差。一方面这是由于调用图的拓扑结构无法像语言分析中那样依靠人类语法、文档结构等知识来设立，所以只能用由左向右模型取代。另一方面，隐马尔可夫一般适用于语言分析等短序列大样本的学习，而对于长度动则上万的系统调用序列则不能得到足够的统计信息。最后，在语言分析问题中，数据往往是人工可以理解并且标注的，然而对于低级的系统调用序列则无法被人工阅读。

结果中很明显的体现了隐马尔可夫级数越高分类效果反而越差的趋势。这也很好的应证了数据稀疏的问题。所以我们可以总结出 HMM 方法在语言分析中非常成功，而在诊断问题上表现却不好的几个原因：一是序列的长度，二是模型拓扑结构的了解程度，三是数据是否可能得到人工标注。

第四，噪声过滤可以很好的提高分类效果，并且比对后的空间上的噪声过滤由于考虑了事件的位置信息，所以效果要优于词频空间上的噪声过滤。有些无关的系统调用，例如对于用户鼠标事件的响应，虽然在所有问题中普遍存在而无法在词频空间上被过滤。然而，在比对后的空间上则因为其出现的不确定性而被滤除。

表 4.5　试验结果比较

Accuracy (%)	**Outlook Express**	**Print**	**Share Folder**	**Use IE**	**Open Word**	**Avg**
Bag of events	84.2	73.3	89.3	80.6	73.2	80.1
Bi-gram	86.1	*75.7	89.3	*81.2	75.1	*81.5
Tri-gram	87.1	71.0	90.5	79.4	*79.1	81.4
Four-gram	*89.6	71.7	90.5	79.2	75.1	81.2
Five-gram	75.2	68.0	*91.9	79.1	74.8	77.8
HMM 5	*75.2	*52.0	*56.0	*56.4	*64.8	*60.9
HMM 10	71.1	47.3	51.7	47.2	58.6	55.2
HMM 15	60.1	48.6	42.7	43.6	58.0	50.6
Alignment	92.9	84.0	93.2	90.2	83.1	88.7
Alignment(A)	92.9	84.2	95.5	90.8	87.7	90.2
Alignment(B)	*95.5	*92.0	*97.7	*90.8	*89.2	*93.0

在运行效率上字符串比对方法也具有其优势。首先相对于 n-gram 等简单特征抽取方法，字符串比对方法虽然需要更长的时间做特征抽取，但是由于字符串比对的时间复杂度与字符串总长度 n 的关系是 n*log(n)，所以花费的时间是完全能够承受的。其次 HMM 这样的序列分类工具一般应用在自然语言句子等长度较小的序列上，而对于长度动则上万的系统调用序列来说运行时间则大大增加。相比之下字符串比对方法则具有更大的运行效率优势。在我们的试验中，

n-gram 与字符串比对方法的学习时间都在 1 分钟以内，而 HMM 学习时间一般都在 6 个小时以上。

为了分析训练样本数对于分类的影响，图 4.7 给出了以不同训练样本大小（电脑数）下的分类效果。我们可以发现，大约 6 个用户以上分类效果就比较稳定了。

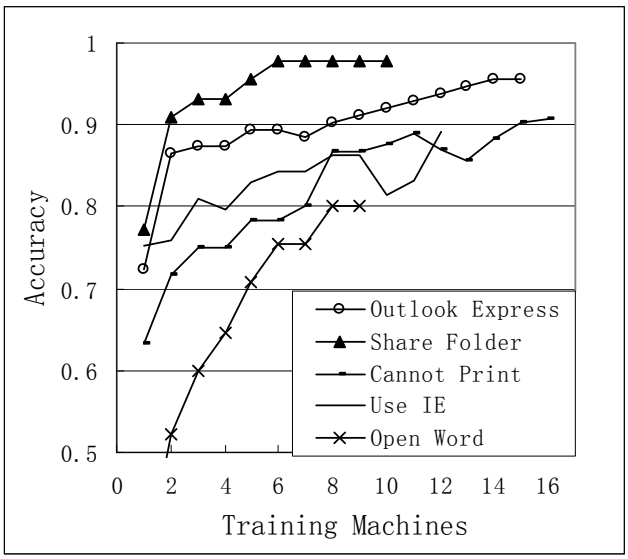

图 4.7　训练样本数量的影响

4.4 讨论

本章介绍了如何收集并利用系统动态事件来进一步提高诊断的精确度和问题覆盖面。基本的思想是使用两层分类器的结构以实现完全无需人工干预的已知问题自动诊断。第一层分类通过错误症状的自动检测实现。第二层通过系统调用序列的分类来实现。这一系统被应用在 Windows 系统的 5 个常见的问题上都取得了良好的效果。

最后我们认为利用系统调用序列推测问题的原因的工作还可以在一下几个方面进行更深入的研究：

1. 计算机系统中是否还有其他类型的信息可以更有效的帮助问题诊断？
2. 什么方法能更有效的从序列数据中抽取特征？
3. 对于大量的数据如何区分出有用的和无用的？如何使得算法有可扩展性？
4. 在真实应用场景下，算法是否可行？

第5章 结论

现代的计算机系统功能越来越强大，结构也越来越复杂,应用数据挖掘的方法解决计算机系统诊断的问题也逐渐成为一个新兴的研究方向。该技术都有很广阔的应用前景。

本文首先简单介绍文计算机自动诊断的研究内容和意义然后概括性地介绍计算机自动诊断的相关工作，包括计算机自动诊断、系统配置管理、通过数据挖掘的方法实现计算机自动诊断等。在介绍的同时，针对提及的算法做了相应的优缺点分析。

本文探索了利用多种系统动态或静态信息来识别已知问题的方法。包括如何利用静态信息实现自动诊断系统，以及如何利用动态事件实现自动诊断系统。其中使用了多种数据挖掘的方法。在真实案例数据上的试验表明这些方法在识别已知问题时可以达到较好的精度。

总结起来讲，本文的贡献主要在于以下四点：

1. 提出并实现了结合高层次的症状描述和低层次的系统信息来提高诊断精确度的方法；
2. 比较了四个上下文无关的查询的概率模型；
3. 提出两层分类器的结构以实现完全无需人工干预的已知问题自动诊断。并且实现了自动检测问题症状的技术，以取代用户对症状的描述；
4. 提出新的基于字符串比对的序列数据分类特征抽取的方法，并给出了翔实的实验来证明其有效性；

本文未来的工作包括：

1. 探索计算机系统中是否还有其他类型的信息可以更有效的帮助问题诊断。
2. 在 3.3 节中，我们主要采用了查询扩展的策略来实现上下文查询模型。接下来的工作中，还可以探索在语言模型(language model)或者翻译模型(translation model)的框架下实现上下文查询的方法[49][50][51][52]。
3. 探索能更有效的从序列数据中抽取特征的方法。
4. 对于第 3 章基于动态信息的诊断，还需要在实际应用的大量数据上检验

算法的可扩展性。

随着相关技术的发展和实际需求的日益增长，自动诊断技术越来越受到人们的关注和重视，相信这是一个极具价值的研究领域，值得更深一步的探索。

参考文献

[1] R. Bace, Intrusion Detection. Macmillan Technical Publishing, 2000

[2] E. Eskin, W. Lee and S. J. Stolfo. ``Modeling System Calls for Intrusion Detection with Dynamic Window Sizes." Proceedings of DISCEX II. June 2001

[3] W. Lee, W. Fan, Mining system audit data: Opportunities and challenges, ACM SIGMOD Record, Volume 30, Number 4 (December 2001), pages 35-44

[4] S. Forrest, S. A. Hofmeyr, A. Somayaji, T. A. Longstaff, A Sense of Self for Unix Processes, Proceedinges of the 1996 IEEE Symposium on Research in Security and Privacy

[5] W. Lee, S. J. Stolfo, P. K. Chan, Learning Patterns from Unix Process Execution Traces for Intrusion Detection Proceedings of the AAAI97 workshop on AI methods in Fraud and risk management

[6] C. Warrender, S. Forrest, and B. Pearlmutter. Detecting intrusions using system calls: Alternative data models. IEEE Symposium on Security and Privacy, 1999

[7] Y. Liao,V. R. Vemuri, Using Text Categorization Techniques for Intrusion Detection11th USENIX Security Symposium, August 5--9, 2002

[8] G. Tandon, P. Chan, D. Mitra, MORPHEUS: Motif Oriented Representations to Purge Hostile Events from Unlabeled Sequences, Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security table of contents, p16 -p25, 200

[9] E. Eskin, W. Lee and S. J. Stolfo. ``Modeling System Calls for Intrusion Detection with Dynamic Window Sizes." Proceedings of DISCEX II. June 2001

[10] D. Gusfield. Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology. Cambridge University Press, January 1997.

[11] M. Russinovich and B. Cogswell. Windows NT System Call Hooking. Dr. Dobb's Journal, January 1997

[12] Microsoft Corporation. Microsoft Knowledge Base, http://support.microsoft.com.

[13] Chun Yuan; Ni Lao; Ji-Rong Wen; Jiwei Li; Zheng Zhang; Yi-Min Wang; Wei-Ying Ma, Automated Known Problem Diagnosis with Event Traces, EuroSys, 2006.

[14] Ni Lao, Ji-Rong Wen, Wei-Ying Ma, Yi-Min Wang, Combine High Level Symptom and Low Level State Information for Configuration Fault Diagnosis, USENIX LISA, pp. 151-158, 2004.

[15] Ji-Rong Wen, Ni Lao, Wei-Ying Ma, Probabilistic Model for Contextual Retrieval, SIGIR, pp. 57-63, 2004

[16] Archana Ganapathi, Yi-Min Wang, Ni Lao, Ji-Rong Wen, Why PCs Are Fragile and What We Can Do About It: A Study of Windows Registry Problems, Dependable System and Network (DSN), pp. 561-566, 2004.

[17] Anderson, E. and D. Patterson, “A Retrospective on Twelve Years of LISA Proceedings,” Proceedings of the Thirteenth Systems Administration Conference (LISA XIII) (Berkeley, CA: USENIX, 1999), p. 95.

[18] Chen, M., E. Kiciman, E. Fratkin, A. Fox, and E. Brewer, “Pinpoint: Problem Determination in Large, Dynamic, Internet Services,” Proc.Int. Conf. on Dependable Systems and Networks (IPDS Track), 2002.

[19] Ganapathi, A., Yi-Min Wang, Ni Lao, and Ji-Rong Wen, Why PCs Are Fragile and What We Can Do About It: A Study of Windows Registry Problems, to appear in Proc. IEEE DSN/DCC, June 2004.

[20] Gray, J., “What Next? A Dozen Information-Technology Research Goals,” Journal of the ACM, Vol. 50, No. 1, January 2003, pp. 41–57.

[21] Larsson, M. and I. Crnkovic, “Configuration Management for Component-based Systems,” Proc. Int. Conf. on Software Engineering (ICSE), May 2001.

[22] Qie, X.-H., Sanjai Narain, “Using Service Grammar to Diagnose BGP Configuration Errors,” in Proc. Usenix Large Installation Systems Administration (LISA) Conference, pp. 237-246 , October 2003.

[23] Redstone, J. A., M. M. Swift, B. N. Bershad, “Using Computers to Diagnose Computer Problems”, Proc. HotOS, 2003.

[24] Solomon, D. A. and M. Russinovich, “Inside Microsoft Windows 2000,” Microsoft Press, 3rd edition, Sept., 2000.

[25] van Rijsbergen, C.J., “Information retrieval,” Butterworths, Second Edition, London, 1979.

[26] Wang, Y.-M., Verbowski, Chad., Dunagan, J., Chen, Y., Wang, H.J., Yuan, C., and Zhang, Z., “STRIDER: A Black-box, State-based Approach to Change and Configuration Management and Support,” in Proc. Usenix Large Installation Systems Administration (LISA) Conference, pp. 159-171, October 2003.

[27] Zipf, G.K., “Human Behavior and Principle of Least Effort: an Introduction to Human Ecology,” Addison Wesley, Cambridge, MA, 1949.

[28] J. Pitkow, H. Sch¨¹tze, T. Cass, R. Cooley, D. Turnbull, A. Edmonds, E. Adar, T. Breuel. Personalized search. Communications of the ACM, v.45 n.9, September 2002

[29] Allan, J. et al, Challenges in Information Retrieval and Language Modeling, Report of a Workshop held at the Center for Intelligent Information Retrieval, University of Massachusetts Amherst, September 2002

[30] Finkelstein, L. et al, Placing Search in Context: The Concept Revisited, In Proceedings of the Tenth International World Wide Web Conference (WWW10), Hong Kong, May 2001

[31] Jin, R., Hauptmann, A. G. and Zhai C., Title Language Model for Information Retrieval, In Proceedings of the ACM SIGIR 2002, pp. 42–48, 2002

[32] Bernard J. Jansen , Amanda Spink , Tefko Saracevic, Real life, real users, and real needs: a study and analysis of user queries on the web, Information Processing and Management: an International Journal, v.36 n.2, p.207-227, Jan.1.2000

[33] Robertson, S. E., Walker, S. and Sparck Jones, M. et, al., Okapi at TREC-3, In D. K. Harman, editor, In Proceedings of the Second Text Retrieval Conference (TREC-3), NIST Special Publication, 500-225, 1995

[34] I. Cohen, S. Zhang, M. Goldszmidt, J. Symons, T. Kelly, and A. Fox. Capturing, Indexing, Clustering, and Retrieving System History. In the 20th ACM Symposium on Operating Systems Principles, October 2005

[35] J.O. Kephart and D.M. Chess. The Vision of Autonomic Computing. IEEE Computer, January 2003

[36] G. Banga. Auto-Diagnosis of Field Problems in an Appliance Operating System. USENIX Annual Technical Conference, San Diego, California, USA, June 2000

[37] M. Chen, A. Accardi, E. Kiciman, A. Fox, D. Patterson, and E. Brewer. Path-based Failure and Evolution Management. USENIX/ACM Symposium on Networked Systems Design and Implementation, San Francisco, CA, March 2004

[38] M. Chen, E. Kiciman, E. Fratkin, A. Fox, and E. Brewer. Pinpoint: Problem Determination in Large, Dynamic Systems. International Conference on Dependable Systems and Networks, IPDS track, Washington, DC, June 2002

[39] P. Barham, A. Donnelly, R. Isaacs, and R. Mortier. Using Magpie for Request Extraction and Workload Modelling. 6th Symposium on Operating System Design and Implementation (OSDI), December, 2004

[40] P. Barham, R. Isaacs, R. Mortier, and D. Narayanan. Magpie: Online Modelling and Performance-aware Systems. 9th Workshop on Hot Topics in Operating Systems, 2003

[41] M.K. Aguilera, J.C. Mogul, J. Wiener, P. Reynolds, and A. Muthitacharoen. Performance debugging for distributed systems of black boxes. In the 19th ACM Symposium on Operating Systems Principles, October 2003

[42] I. Cohen, J. Chase, M. Goldszmidt, T. Kelly, and J. Symons. Correlating Instrumentation Data to System States: A Building Block for Automated Diagnosis and Control. 6th Symposium on Operating Systems Design and Implementation (OSDI '04), December 2004

[43] S. Hofmeyr, S. Forrest, and A. Somayaji. Intrusion Detection Using Sequences of System Calls. Journal of Computer Security, Vol. 6, pp. 151-180, 1998

[44] W. Lee and S. Stolfo. Data Mining Approaches for Intrusion Detection. In Proceedings of the Seventh USENIX Security Symposium, San Antonio, TX, January 1998

[45] C. Warrender, S. Forrest, and B. Pearlmutter. Detecting intrusions using system calls: Alternative data models. IEEE Symposium on Security and Privacy, 1999

[46] F. Apap, A. Honig, S. Hershkop, E. Eskin, and S.J. Stolfo. Detecting Malicious Software by Monitoring Anomalous Windows R egistry Accesses. In Proceedings of the Fifth International Symposium on Recent Advances in Intrusion Detection (RAID-2002). Zurich, Switzerland, October 2002

[47] Lawrence, S., Context in Web Search, IEEE Data Engineering Bulletin, Volume 23, Number 3, pp. 25–32, 2000

[48] Finkelstein, L. et al, Placing Search in Context: The Concept Revisited, In Proceedings of the Tenth International World Wide Web Conference (WWW10), Hong Kong, May 2001

[49] Berger, A. and Lafferty, J., Information Retrieval as Statistical Translation. In Proceedings of ACM SIGIR 1999, pp. 222-229, 1999

[50] Jin, R., Hauptmann, A. G. and Zhai C., Title Language Model for Information Retrieval, In Proceedings of the ACM SIGIR 2002, pp. 42–48, 2002

[51] Lafferty, J. and Zhai, C., Document Language Models, Query Models, and Risk Minimization for Information Retrieval, In Proceedings of the ACM SIGIR 2001, pp. 111–119, 2001

[52] Ponte, J. and Croft, W. B., A Language Modeling Approach to Information Retrieval. In Proceedings of the ACM SIGIR 1998, pp. 275–281, 1998

[53] Vapnik, V. N. 1995. The nature of statistical learning theory. Berlin: Springer-Verlag.

[54] Gunn, S. R. 1997. Support vector machines for classification and regression, Technical Report. Image Speech and Intelligent Systems Research Group, University of Southampton, Southampton.

[55] S. T. Dumais, J. Platt, D. Heckerman and M. Sahami. Inductive learning algorithms and representations for text categorization. In Proceedings of ACM-CIKM98, Nov. 1998, pp. 148-155

[56] C. Y. Suen, “N-Gram Statistics for Natural Language Understanding and Text Processing,” IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. PAMI-1, No. 2, April 1979, pp.164-172.

[57] W. B.Cavnar, J. M. Trenkle, "n-gram Based Text Categorization, Proceedings of the Third Annual Symposium on Document Analysis and Information Retrieval", April

1994, pp 161-169

[58] L. R. Rabiner, A tutorial on hidden Markov models and selected applications in speech recognition, Proceedings of the IEEE, Vol. 77, No. 2, February 1989

[59] D. Bikel, S. Miller, R. Schwartz, R. Weischedel, Nymble: a high-performance learning name-finder. Fifth Conference on Applied Natural Language Processing, (published by ACL), pp 194-201 (1997).

[60] J. Makhoul, R. Schwartz, C LaPre, I. Bazzi, A script-independent methodology for optical character recognition. Pattern Recognition, Vol 31, No. 9, pp. 1285-1294 (1998).

[61] K. Seymore, A. McCallurn, R. Rosenreid, "Learning hidden Markov model structure for information extraction", In Proceedings of the AAAI-99 Workshop on Machine Learning for Information Extraction, pp 37-42, 1999.

[62] E. Eskin, W. Lee and S. J. Stolfo. ``Modeling System Calls for Intrusion Detection with Dynamic Window Sizes." Proceedings of DISCEX II. June 2001

[63] P. Smyth, D. Heckerman, and M. Jordan. Probabilistic independence networks for hidden Markov probability models. Neural Computation, 9(2):227-269, 1997.

[64] L. R. Rabiner, A tutorial on hidden Markov models and selected applications in speech recognition, Proceedings of the IEEE, Vol. 77, No. 2, February 1989

致　谢

感谢我的导师李春平老师，给予我悉心的指导，使我能够顺利完成硕士论文。李老师严谨的治学态度、平易近人的学者风范，已使我受益匪浅。学业上的谆谆教导，生活上的关心，令人感动。

感谢微软亚洲研究院的文继荣研究员。本论文中大部分工作都在与他讨论中完成。他对研究的热情和深刻理解使我受益非浅。

最后感谢在我撰写这篇论文中给予我细心帮助的谌卫军老师。同时感谢我的女友左闻韵给予的支持。

声　明

本人郑重声明：所呈交的学位论文，是本人在导师指导下，独立进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。

签　名：　　日　期：

个人简历、在学期间发表的学术论文与研究成果

个人简历

1980 年 10 月 24 日出生于贵州省贵阳市。

1999 年 9 月考入清华大学电子工程系。

2003 年 9 月免试进入清华大学软件学院攻读软件与理论专业硕士至今。

发表的学术论文

[1] Chun Yuan; Ni Lao; Ji-Rong Wen; Jiwei Li; Zheng Zhang; Yi-Min Wang; Wei-Ying Ma, Automated Known Problem Diagnosis with Event Traces, EuroSys, 2006.

[2] Ni Lao, Ji-Rong Wen, Wei-Ying Ma, Yi-Min Wang, Combine High Level Symptom and Low Level State Information for Configuration Fault Diagnosis, USENIX LISA, pp. 151-158, 2004.

[3] Ji-Rong Wen, Ni Lao, Wei-Ying Ma, Probabilistic Model for Contextual Retrieval, SIGIR, pp. 57-63, 2004

[4] Archana Ganapathi, Yi-Min Wang, Ni Lao, Ji-Rong Wen, Why PCs Are Fragile and What We Can Do About It: A Study of Windows Registry Problems, Dependable System and Network (DSN), pp. 561-566, 2004.

研究成果

美国专利：

Method and System for Troubleshooting a Misconfiguration of a Computer System Based on Product Support Services Information. 41826.8030US