

国内图书分类号: TN915.08

密级: 公开

国际图书分类号: 621.39

# 西南交通大学 研究生学位论文

## PDF 文档解析与内容脱敏技术研究

年 级 2015

姓 名 朱玲玉

申请学位级别 硕士

专 业 信息与通信工程

指 导 教 师 陈庆春

二零一八年五月九日

Classified Index: TN915.08

U.D.C: 621.39

Southwest Jiaotong University

Master Degree Thesis

# PDF DOCUMENT PARSING AND CONTENT DESENSITIZATION TECHNIQUES

Grade:2015

Candidate:Lingyu Zhu

Academic Degree Applied for : Master Degree

Speciality:Information&Communication Engineering

Supervisor:Qingchun Chen

May.9,2018

# 西南交通大学

## 学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权西南交通大学可以将本论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复印手段保存和汇编本学位论文。

本学位论文属于

1. 保密□，在 年解密后适用本授权书；
2. 不保密☑，使用本授权书。

(请在以上方框内打“√”)

学位论文作者签名：朱玲玉

指导老师签名：陈乐春

日期：2018-5-14

日期：2018-5-14

## 西南交通大学硕士学位论文主要工作(贡献)声明

本人在学位论文中所做的主要工作或贡献如下:

1. 深入研究 PDF 文件格式以及传统 PDF 文件的解析方法, 为了支持对网络中的 PDF 文件进行在线处理, 论文给出了一种基于 Stream 流的在线 PDF 文件解析处理方法。
2. 在对 PDF 文件文本内容进行解析的基础上, 为快速、准确地定位脱敏内容, 论文在 PDF 文本内容编码的基础上, 提出一种适用于 PDF 文档文本内容的高效内容匹配算法。
3. 在反向代理机制框架下, 论文给出了一种基于网络 PDF 文件识别、敏感内容识别定位与敏感内容脱敏处理的 PDF 文档解析与内容脱敏技术解决方案。

本人郑重声明: 所呈交的学位论文, 是在导师指导下独立进行研究工作所得的成果。除文中已经注明引用的内容外, 本论文不包含任何其他个人或集体已经发表或撰写过的研究成果。对本文的研究做出贡献的个人和集体, 均已在文中作了明确说明。本人完全了解违反上述声明所引起的一切法律责任将由本人承担。

学位论文作者签名: 朱玲玉

日期: 2018-5-14

## 摘 要

随着互联网的飞速发展以及大数据时代的来临，人们获取信息资源的手段也更加便捷，同时也带来了信息泄漏、敏感信息传播等问题。如何在保证数据资源开放及共享的条件下，防止涉密信息内容泄漏成为网络内容安全管控技术亟待解决的一大关键问题。PDF 文档作为目前应用最为广泛的一种电子文档，涉及 PDF 文档内容防控的相关研究工作具有重要的应用价值。现有的 PDF 文档解析工具大多数是对本地的 PDF 文件进行解析，而较少能对网络在线 PDF 文档进行解析处理；其次，目前针对 PDF 文档内容涉密内容脱敏处理的工具更少。考虑到未来网络中 PDF 电子文档的广泛传播，应用中确实也存在对 PDF 文档内容中用户指定的敏感信息内容扩散的防控要求，论文重点研究在线 PDF 文档解析与内容脱敏技术。

论文首先在概述当前网络信息安全所面临的挑战和技术解决方案的基础上，论证了对电子文档中敏感信息脱敏处理的必要性。然后在对当前 PDF 文件解析技术和存在问题分析的基础上，提出了一种基于 Stream 流的 PDF 文件解析方法，该方法不仅适用于本地 PDF 文件的处理，还适用于对网络中 PDF 文档的实时在线处理。

在对 PDF 文件解析的基础上，为了更高效的对用户设定的敏感信息进行确认和处理，论文在对经典的字符匹配算法进行分析和比较的基础上，综合考虑 BM 算法和 QS 算法的优点，结合 PDF 文件文本编码特征，研究并给出了一种高效的字符匹配改进算法。有关实验验证结果表明，新算法能有效改进匹配效率。

最后，在反向代理机制框架下，论文给出了一种基于网络在线的 PDF 文件识别、敏感内容识别定位与敏感内容脱敏处理的 PDF 文档解析与内容脱敏技术解决方案，并进行了功能测试和系统性能测试。实验结果表明，论文所完成的网络在线的 PDF 文件识别、敏感内容识别定位与敏感内容脱敏处理具有良好的 PDF 文档内容脱敏效果，可以满足实际应用要求。

论文的相关分析工作为后续深入研究高效的网络电子文档内容防控技术具有一定的参考和借鉴价值。

**关键词：**内容防控；内容解析；内容脱敏；PDF 文档

## Abstract

With the rapid development of the Internet and the dawn of big data era, the way people access information resources becomes much more convenient, but it also brings about problems like unwanted information release and dissemination of sensitive information. Under the condition of ensuring the openness and sharing of data resources, how to prevent the unwanted disclosure of confidential information has becomes a key issue that needs to be solved carefully. As a widely used electronic document, the related research work on prevention and control of PDF document content find potential applications. Most of the existing PDF document parsing tools can handle only the local (offline) PDF files, few is able to deal with online PDF documents. Secondly, there are few tools for the desensitization of confidential messages in PDF document. Considering the widespread dissemination of PDF electronic documents in future networks, there is indeed a urgent need for prevention and control of the proliferation of user-specified sensitive information in the contents of PDF documents. The paper focuses on the analysis of online PDF document resolution and the related content desensitization techniques.

Firstly, based on the brief survey of challenges faced by current network information security and technical solutions, the thesis demonstrates the necessity of desensitization of sensitive information in electronic documents. Then, based on the analysis of the current PDF file parsing technology and existing problems, this thesis proposes a PDF file parsing framework. This method is not only suitable for local PDF files, but also applicable to the online PDF documents in the network.

On the basis of parsing the PDF file, in order to efficiently locate and confirm the confidential message specified by the user, motivated by the analysis and comparison of the classical matching algorithm of both BM algorithm and QS algorithm are addressed. In this thesis, an efficient keyword matching algorithm is proposed by considering the features of PDF file text content encoding. The experimental verification results show that the new algorithm can effectively improve the matching efficiency.

Finally, within the framework of reverse proxy mechanism, this thesis presents a solution of PDF document parsing and content desensitization based on on-line PDF file identification, confidential content identification location and the related content desensitization processing. Function test and system performance test are also carried out. The experimental results show that, the online PDF file recognition, confidential content identification, and confidential content desensitization processing can meet the practical

---

application requirements.

The related analysis work of the paper has certain reference value for the further study of the effective network electronic document content control technology.

**Key words:** content prevention and control; content analysis; content desensitization; PDF document

<http://www.ixueshu.com>

---

# 目 录

第 1 章 绪 论.....	1
1.1 课题研究背景及意义.....	1
1.2 国内外研究现状及存在问题.....	2
1.3 论文主要工作及内容安排.....	4
第 2 章 PDF 文档格式.....	5
2.1 PDF 对象.....	5
2.1.1 PDF 对象的基本类型.....	5
2.1.2 对象分类.....	6
2.2 文件（物理）结构.....	7
2.2.1 文件头.....	7
2.2.2 文件体.....	8
2.2.3 交叉引用表.....	8
2.2.4 文件尾.....	9
2.3 逻辑结构.....	10
2.3.1 文档目录.....	11
2.3.2 页树.....	11
2.3.3 页节点.....	12
2.4 内容流.....	13
2.4.1 Stream 流的解/压缩方法.....	13
2.4.2 解压后的内容流.....	13
2.5 中文的转码.....	14
2.5.1 Cmap.....	14
2.5.2 字符码转为 unicode 码.....	16
2.5.3 ToUnicode Cmap 的转换.....	16
2.6 本章小结.....	18
第 3 章 PDF 文档解析及脱敏处理.....	19
3.1 PDF 文档内容解析.....	19
3.2 基于 Stream 流的 PDF 文档解析.....	22
3.2.1 PDF 文件识别.....	22
3.2.2 分段解压及封装.....	22
3.2.3 Stream 流信息的分类.....	24
3.2.4 萃取 Cmap 流.....	24



3.2.5 正文及注释的提取 .....	25
3.3 脱敏操作 .....	26
3.4 PDF 文件脱敏流程及实现 .....	27
3.4.1 PDF 文件整体脱敏流程 .....	27
3.4.2 主要类和函数功能实现 .....	30
3.5 本章小结 .....	32
第 4 章 模式匹配算法 .....	33
4.1 经典单模式匹配算法 .....	33
4.1.1 相关概念介绍 .....	33
4.1.2 BM 算法 .....	33
4.1.3 QS 算法 .....	36
4.2 改进算法 .....	38
4.2.1 算法思想 .....	38
4.2.2 算法预处理及匹配过程 .....	38
4.2.3 算法实现 .....	39
4.2.4 算法实例 .....	39
4.2.5 测试结果分析 .....	41
4.3 本章小结 .....	46
第 5 章 脱敏系统的框架设计与实现 .....	47
5.1 HTTP 协议 .....	47
5.1.1 HTTP 协议简介 .....	47
5.1.2 HTTP 报文请求格式 .....	48
5.1.3 HTTP 报文响应消息格式 .....	48
5.2 总体架构 .....	50
5.3 基于反向代理的脱敏系统模块的设计与实现 .....	52
5.3.1 TCP 反向代理模块 .....	52
5.3.2 HTTP 协议解析模块 .....	53
5.3.3 内容解析模块 .....	55
5.3.4 敏感词控制模块 .....	55
5.4 实验结果分析 .....	57
5.4.1 实验环境搭建 .....	57
5.4.2 脱敏效果测试 .....	57
5.4.3 系统性能分析 .....	59
5.5 本章小结 .....	61

---

结    论.....	62
致    谢.....	64
参考文献.....	65
攻读硕士期间发表的论文及科研成果.....	69

---

<http://www.ixueshu.com>

---

# 第 1 章 绪 论

## 1.1 课题研究背景及意义

互联网的快速发展使网络越来越普及,截止2018年1月,在中国互联网发展状况的第41次统计报告中显示,中国网民数已达7.72亿,占全球互联网用户总数的1/5,互联网普及率55.8%,超全世界平均水平的4.1个百分点<sup>[1]</sup>。

网络的普及化及大数据时代的来临,人们获取和共享数据资源也越来越便捷。但互联网的迅速普及也带来了一些新问题,如网络易受到入侵,企业隐私信息容易泄漏,给企业带来损失等。2015年2月,一场网络攻击同时入侵世界共计100家银行系统,造成了巨大的损失;同年一月,中国铁通被爆系统漏洞,造成用户信息和商业合作的信息泄露<sup>[1]</sup>。信息通信网络与安全规划(2016-2020)强调,在网络技术大力发展的情况下,亟待加强对通信网络数据和用户信息的保护,建立并完善数据和信息泄露安全机制<sup>[2]</sup>。一般通信网络信息安全存在的主要问题为信息的滥用、信息的篡改、以及信息的窃取等<sup>[3]</sup>。目前,国内外针对数据信息安全,大概采取如下技术:如防火墙技术、入侵检测技术、身份认证技术、信息加密技术以及漏洞扫描技术等<sup>[4]</sup>。以往的文件资料就是按传统的纸质文件保存,中间同样会存在个人信息和敏感信息的泄漏问题,比如一些银行卡号,身份证号等信息。显然,敏感文件资料的纸介质保密管理措施已经不能适应电子文档保密规定的要求以及资源的共享特性<sup>[5]</sup>。随着信息技术的发展、办公自动化以及国家提倡的网络信息化建设,电子文件逐渐取代了纸质文档;而针对带有敏感信息的电子文档的内容防控成为一个亟待解决问题。

一般而言,敏感信息是指其丢失、不当使用或未经授权被人接触或修改会不利于国家利益或不利于个人依法享有的个人隐私权的所有信息<sup>[6]</sup>。在云化的网络内容共享平台下,一些敏感信息可能对内或对一定级别以上人员是无需保密处理的,但对外部或内部级别较低人员而言是需要进行内容防控的。电子文档存在容易被复制、传播速度快等特点,处理不当,容易导致敏感信息内容泄密<sup>[7]</sup>。因此对电子文档的敏感信息的保护,或者防止非有效用户的访问和处理,或对电子文档的敏感数据进行数据脱敏是有应用价值。

作为一类应用比较多的电子文档,PDF文档具有独立于创建它的应用软件、硬件、操作系统、显示或打印它的输出设备,可有效保留原始文档的外观和完整性,还具有跨平台,自由共享特点,被广泛应用于电子文档的信息存储与传递。从1992年至今,PDF文档从最初的1.0版本规范发展到了1.7版本,其存储信息的类型也变得多样化,覆盖了文字、图形、图像、声音和动画等类型,还支持对文本、超链接、标记、声音、视频的注释。PDF可移植文档格式已于2008年被ISO批准为国际开放标准。目前,网络

和大型数据库中的电子文档大多数是PDF文档。因此对于PDF文档内容的解析和敏感信息的脱敏处理技术研究显得尤为重要。

大数据时代的到来，带来了机遇同时也面临挑战。目前，各大行业致力于数据挖掘的研究，并获得商机；与此同时，为避免电子文档中所包含的某些敏感信息被传播或非法利用，维护网络信息安全，需要采用切实有效的技术根据需要对电子文档进行数据脱敏处理。数据脱敏是指某些敏感信息通过脱敏规则进行数据的变形，实现敏感隐私数据的可靠保护。在现实应用中，凡涉及客户安全数据或者一些商业性敏感数据，在不违反系统规则条件下，往往需要对真实的身份证号、手机号、卡号、客户号等个人信息数据进行脱敏处理<sup>[8]</sup>。

传统的 PDF 文件解析方法一般针对的是本地的 PDF 文件，为了支持 PDF 文档在网络在线状态下完成内容脱敏，需要解决 PDF 文档识别和截获、PDF 文档内容解析与敏感信息匹配，以及敏感信息的脱敏处理<sup>[9]</sup>，而现有的 PDF 文档分析工具大都不具有这些能力，这也正是本论文研究最为重要的一个研究出发点。

## 1.2 国内外研究现状及存在问题

### 1. 数据脱敏（Data Masking）研究现状

为维护通信网络安全，避免信息的滥用、涉密信息的窃取、篡改，国内外针对数据脱敏做了一定的研究。Adam 和 Wortmann 较早研究了数据脱敏问题<sup>[10]</sup>。现有的数据脱敏相关研究大多是围绕脱敏算法来展开的。文献[11]在传统脱敏算法的基础上，提出了归一化算法。在保证数据价值的前提下，文献[12]提出了数据变形和比特位变形技术。除了脱敏算法，国内外围绕数据脱敏技术框架也展开了大量研究，文献[13]总结了 Oracle 提出的脱敏模型和 IBM 提出的 OPTIM 脱敏系统，以及在两者的基础之上的通用数据脱敏模型。基于保形加密文献[14]提出了一种大数据脱敏系统。文献[15]分析和探讨了非结构化数据的内容识别问题。文献[16]在动态脱敏的环境下，分别对基于视图的脱敏机制和基于代理的脱敏机制，提出了大数据环境下的数据脱敏框架，该框架支持对文本数据的数据脱敏处理。

### 2. PDF 文件解析研究现状

国内外围绕 PDF 文档解析的研究主要集中于如何将 PDF 文档转换为其他文件格式、PDF 文档中特殊格式的识别等，也有一些研究专注于 PDF 文档中的图形图像处理、内容的组织和标签内容等<sup>[17]</sup>。专门针对 PDF 文档内容解析的相关研究还比较少。

一些应用软件提供了一些 PDF 类库，例如基于 java 的 PDFBox 和 iText、基于 C++ 的 XPDF、基于 .Net 的 iTextSharp、PDFlib 等<sup>[18]</sup>，但这些库大多用于生成 PDF 文档。专门用于 PDF 解析的类库比较少，并且或多或少存在一些问题，例如无法很好地支持中文、只支持特定版本的 PDF 文档。而且这些 PDF 类库大都是封装好的，一般只支持

PDF 文档中文本提取和显示，而不支持对 PDF 文档内容直接进行修改编辑。

文献[19]探讨了如何利用 PDFBox 来提取 PDF 文件中的文本内容，但 PDFBox 对中文内容的支持不好，只能处理原义字符文本流。文献[20]利用 iText 提供的 PdfReader 类研究如何将 word 文档转换 PDF 格式文件。李贵林等人探讨了如何利用 Adobe 提供的 plug-in 插件完成 PDF 文件处理<sup>[21]</sup>。文献[22]介绍了如何利用 PDF2TXT 软件、PDF2HTML，把 PDF 文档转化为其他格式，然后再提取文本内容的技术思路，但是这个技术途径同样存在无法修改 PDF 文档中文本内容的问题。

### 3. 字符串匹配研究现状

作为一类基础算法，国内外研究并给出了许多字符串匹配算法。范洪博在文献[23]中概述和总结了单模式字符串匹配算法，并提出了改进算法。姚期智在文献[24]中证明了单模式匹配算法的最优时间复杂度为  $O(n/m)$ 、最差时间复杂度为  $O(n)$ 。研究结果表明，BM 算法<sup>[25]</sup>和 MP 算法<sup>[26]</sup>分别对应着最优和最差这两种情况。

作为一类最基本的字符串匹配算法，BF 算法为了避免字符回溯，衍生出线性匹配算法 MP，在此基础上，进一步发展出了基于前缀匹配的 KMP<sup>[27]</sup>算法。1997 年出现了第一个基于后缀匹配的 BM 算法，使平均时间复杂度可达到亚线性，成为所有基于后缀匹配算法的基础。在此基础上，文献[28]中提出了 BM 的简化版 Horspool 算法。文献[29]中将 Horspool 和 KMP 结合得到了 FJS 算法。考虑字符串最大跳跃距离、字符出现的频率以及坏字符出现的组合，分别发展出了 QS<sup>[30]</sup>算法、TunBM<sup>[31]</sup>算法和 BR<sup>[32]</sup>算法。在 QS 算法的基础上，SSABS 引入了当前窗口字符的匹配顺序，提高了匹配性能<sup>[33]</sup>。文献[34]基于 BR 算法和 SSABS 算法提出了 TVSBS 算法，适用于短模式串的字符匹配算法。在单模式字符串匹配算法的基础上，后续的很多研究大都是在经典的 KMP、BM、BMH 等算法上的改进，并应用于不同的场景，如围绕字符串的匹配顺序，已有研究分析和考虑了从左往右、从右往左和先右、左再中间的匹配顺序<sup>[35-36]</sup>。此外还有部分研究从移位表方面进行改进，例如考虑匹配窗口的下 2 个字符移位表、比较失配字符前面 2 个字符和下一字符的移位表来选择最大、比较失配字符到下一字符的移位表来选择最大等<sup>[37-38]</sup>。

为了有效支持 PDF 文档内容敏感信息的脱敏处理，我们需要同时考虑 PDF 文件内容解析和敏感词字符串的高效匹配。除了 Adobe 公司提供的开源的 SDK 可以用于 PDF 文件内容解析之外，现有的其他解析方法或软件大都只能对某些特殊的 PDF 文档进行解析处理<sup>[39]</sup>。而那些解析效果比较好的工具其解析过程对用户来说都是不透明的。此外，现有的大部分的 PDF 解析工具或软件只能处理本地 PDF 文件，而较少支持对网络中的 PDF 文件进行实时处理。为此，我们必须自行开发适合于网络应用要求的 PDF 文档内容解析工具。此外，为了支持高效的敏感信息内容的脱敏处理，需要有高效的敏感信息或敏感字符串的确认手段，虽然有大量经典的字符串匹配算法可以用于实现

PDF 文档中敏感信息的定位和确认，但鉴于 PDF 文档特殊的编码格式，直接应用这些算法的字符匹配性能并不好，还需要开发适合于 PDF 特殊编码格式的敏感字符串匹配算法。

### 1.3 论文主要工作及内容安排

论文主要在对当前 PDF 文件解析技术和存在问题分析的基础上，提出了一种基于 Stream 流的 PDF 文件解析方法，该方法不仅适用于本地 PDF 文件的处理，还适用于对网络中 PDF 文档的实时在线处理。

其次对 PDF 文件解析的基础上，为了更高效的对用户设定的敏感信息进行确认和处理，在对经典的字符匹配算法进行分析和比较的基础上，综合考虑 BM 算法和 QS 算法的优点，结合 PDF 文件文本编码特征，研究并给出了一种高效的字符匹配改进算法。有关实验验证结果表明，新算法能有效改进匹配效率。

最后，在反向代理机制框架下，论文给出了一种基于网络在线的 PDF 文件识别、敏感内容识别定位与敏感内容脱敏处理的 PDF 文档解析与内容脱敏技术解决方案，并进行了功能测试和系统性能测试。实验结果表明，论文所完成的网络在线的 PDF 文件识别、敏感内容识别定位与敏感内容脱敏处理具有良好的 PDF 文档内容脱敏效果，可以满足实际应用要求。

论文内容安排如下：

第一章首先介绍了本文的研究背景及意义，对当前网络信息安全面临的挑战、PDF 文件解析技术和存在的问题、字符匹配算法进行分析，论证对应用广泛的 PDF 文档的敏感信息处理的必要性。

第二章主要深入研究 PDF 文档的格式，主要从其对象、逻辑结构、物理结构、文本内容编码方式、内容流进行介绍。

第三章主要对当前 PDF 文档解析技术进行分析，提出一种基于 Stream 流的 PDF 文档的解析方法，该方法不仅适用于本地的 PDF 文件处理，还适用于网络中 PDF 文件的在线处理。

第四章主要对经典字符匹配算法进行分析，考虑 BM 和 QS 算法的优点，结合 PDF 文本内容的编码特征，研究并给出了一种高效的字符匹配改进算法，并验证了其有效性。

第五章主要在反向代理机制框架下，给出对网络在线 PDF 文件的解析及内容脱敏方案，并进行功能测试和系统性能测试，得出该方案具有良好的 PDF 文档内容脱敏效果。

## 第 2 章 PDF 文档格式

PDF 文档格式和 HTML、办公文档、XML 格式化文件一样，有关键字、文本内容、控制信息等。但与其他文档所不同的是，PDF 文档是 8-bit 字节序，以二进制流保存，而 HTML 以文本方式保存。本章将深入研究 PDF 文档格式，主要从 PDF 文件对象、文件结构（逻辑结构）、文档结构（物理结构）、内容流、编码及转换方式方面来详细介绍 PDF 文档格式，为后续的 PDF 文档内容解析分析奠定基础。

### 2.1 PDF 对象

PDF 文档是由一系列对象组成的数据结构，该对象都是一些基本数据类型。PDF 对象可分为直接对象和间接对象。此外，对象可以被标记，以便它们可以被其他对象引用，被标记的 PDF 对象称为间接对象，一旦 PDF 对象被标识，就不再属于八种类型中任一直接对象，而变成了间接对象，可以被别的 PDF 对象引用。

#### 2.1.1 PDF 对象的基本类型

PDF 支持以下八种基本类型的对象<sup>[40]</sup>：逻辑(Boolean)、数值(Numeric)、字符串(String)、名字(Name)、数组(Array)、字典(Dictionary)、流对象(Stream)、空对象(Null)，具体介绍如表 2-1 所示。

表 2-1 基本对象类型表

对象类型	描述
逻辑对象	标识：ture 和 false；可用作数组（Array）元素和字典项的值
数值对象	支持 2 种类型：整数和实数，整数如 55、666、-777 等，实数以定点小数表示，如 3.12、-0.77
字符串对象	由一系列字节组成，范围是 0-255 间无符号整数，长度小于 65535；两种表现形式：原义字符序列(PDF )和十六进制数据<2B09086F>
名称对象	由一个斜杠“/”和一系列原义字符组成，且之间无空白符；长度小于 127，区分大小写，如/b 和/B；PDF1.2 后，名称对象的形式可是“#dd”，（dd 为 2 位十六进制数），如/Adobe#20PDF 代表 Adobe PDF
数组对象	顺序排列对象的一维集合，其元素可以为多种类型的对象，如[666 3.17 ture /myname (hello)]，长度小于 8191

表 2-1 基本对象类型表（续）

对象类型	描述
字典对象	PDF 文档中主要的构造块，是成对对象的关联表，以键值对的方式出现在包含在一对双尖括号<<>>，如图 2-1 所示
流对象	流对象和字符串对象相似，是一个字节序列，可以递增方式读取流对象，流对象长度不受限制。因此，流对象中含大量数据，如图片、页面描述，如图 2-2 所示
空对象	空对象只有一个类型和值，标识：null

其中字典对象格式如图 2-1 所示。

```
15 0 obj
<</Contents [16 0 R 17 0 R]
/MediaBox [0 0 612 825.12 ]
/Parent 1 0 R
/Resources<<
/ProcSet [/PDF /Text /ImageB /ImageC /ImageI ] >>
/Type /Page >>
endobj
```

图 2-1 字典对象示意图

流对象由一个字典、关键字 stream 和 endstream 中间的内容组成，如图 2-2 所示。

```
16 0 obj
<</Type/ObjStm/11 15/First 104/Filter/FlateDecode/Length 485>>
Stream
.....流的内容.....
endstream
endobj
```

图 2-2 流对象示意图

流对象必须是一个间接对象，而其字典必须是直接对象。在关键字 stream 和 endstream 之间存放流内容，其内容可能为文本内容、Cmap 转码映射文件等。流内容的其他相关属性在流字典中规定，如表 2-2 所示。

表 2-2 主要字典属性表

Key	Value
Length	Stream 和 endstream 之间流内容的长度，若有 filter，指经压缩后的算法
Filter	该属性指流内容采用的压缩算法

## 2.1.2 对象分类

### 1. 直接对象

PDF 由一系列对象构成，其直接对象包含 2.1.1 介绍的 8 种基本类型。



## 2. 间接对象 (Indirect)

PDF 文件中任何对象都可以被标记, 成为一个间接对象, 使该对象有一个标识符, 方便被其他对象引用, 比如该对象作为一个数组元素或字典的某个键值对的值。

间接对象被对象号和生成号惟一标识, 在整个生存周期内, 即使该对象的值改变, 其对象号和生成号也保持不变。间接对象由对象号、生成号、以及紧跟其后的关键字 “obj” 和 “endobj” 之间的对象构成, 如下所示:

```
10 0 obj
(2018)
endobj
```

在 PDF 文件的其它位置, 以上原义字符串对象就可以被引用, 通过对象号、生成号、关键字 “R” 可对间接对象进行引用, 如 10 0 R。

## 2.2 文件 (物理) 结构

PDF 文件由对象构成, 高效地访问、增量更新等操作需要很好地理解文档的物理结构。PDF 文件从大的方面来说可分为如图 2-3 所示的 4 个部分:

- 1) 文件头 (Head), 在 PDF 文档的第一行, 标识 PDF 规范的版本号;
- 2) 文件体 (Body), PDF 文件的主要部分, 由一系列对象组成;
- 3) 交叉引用表 (Xref), 包含所有间接对象的地址信息, 以便快速访问;
- 4) 文件尾, 记录交叉引用表的地址, 以及特殊对象 (Catalog) 的位置。



图 2-3 PDF 文件物理结构图

### 2.2.1 文件头

PDF 文件的第一行指明该文件 PDF 规范版本号, 比如 PDF1.5 版的文件, 头部应为: %PDF-1.5。如图 2-4 所示。

```
1  %PDF-1.5 ←
2  %DÔÅØ
3  3 0 obj <<
4  /Length 401
5  /Filter /FlateDecode
6  >>
7  stream
8  xÚmRK>0DLE%çWøhK...ÅDLERS9UjÕJÝ3·jSI50ž
```

图 2-4 文件头标识示意图

### 2.2.2 文件体

文件体是由一系列间接对象组成，这些对象可能代表文档的内容及其相关信息，如文本、图像等。从 PDF1.5 开始，文件体中也可以是对象流，对象流由一系列对象组成，如图 2-5 所示。

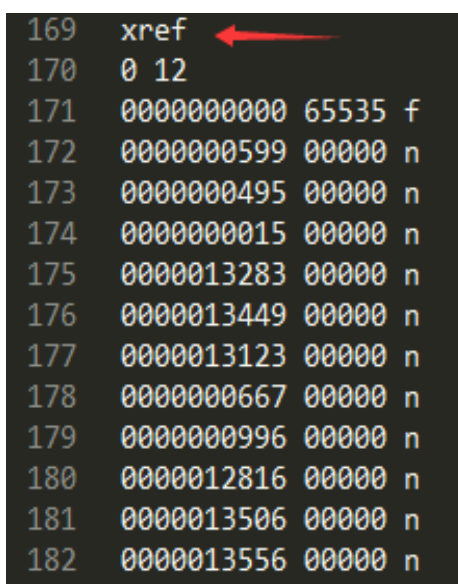
```
3  3 0 obj << ←
4  /Length 401
5  /Filter /FlateDecode
6  >>
7  stream
8  xÚmRK>0DLE%çWøhK...ÅDLERS9UjÕJÝ3·j
9  ŸÄÜd2À}BELL÷ŠŸ|SYN“Ú+ETBStëu%‘!-DC4Ö
10 endstream
11 endobj
12 2 0 obj << ←
13 /Type /Page
14 /Contents 3 0 R
15 /Resources 1 0 R
16 /MediaBox [0 0 612 792]
17 /Parent 5 0 R
18 >> endobj
19 1 0 obj << ←
20 /Font << /F51 4 0 R >>
21 /ProcSet [ /PDF /Text ]
22 >> endobj
23 7 0 obj ←
24 [250 333]
25 endobj
26 8 0 obj << ←
```

图 2-5 文件体示意图

### 2.2.3 交叉引用表

交叉引用表起索引作用，由一些地址信息构成，通过这些地址偏移信息就可以随机地访问文件中的某些对象，而不用读完整个文件来设立地址索引表。在交叉引用表中，每一个间接对象有对应的一行项，指出间接对象在文件中的位置，以便进行解析、修改、增加等。从 PDF1.5 版本开始，和对象流类似，交叉引用表也可能出现在交

又应用流中为便于每一项能被随机访问，交叉引用表每一项都有固定的格式，如图 2-6 所示。



169	xref
170	0 12
171	0000000000 65535 f
172	0000000599 00000 n
173	0000000495 00000 n
174	0000000015 00000 n
175	0000013283 00000 n
176	0000013449 00000 n
177	0000013123 00000 n
178	0000000667 00000 n
179	0000000996 00000 n
180	0000012816 00000 n
181	0000013506 00000 n
182	0000013556 00000 n

图 2-6 交叉索引表示意图

交叉引用表由关键字“xref”和随后的一系列交叉引用子项构成，且子项的对象计数从 0 开始。如果一个 PDF 文件从未更新过，只有一项，其对象计数以 0 开始。子项最前面有 2 个数字，分别代表其子项的第一个对象号和子项的个数。如：0 12 代表一个以对象号 0 开始，编号从 0 到 11 共有 12 个对象，每一项包括行结束符有 20 字节。

另外，交叉引用项分为 2 类，格式相同但意义不同，一类为正在使用对象（有效项 n），另一类为已被删除的对象（空闲项 f）。

子项格式：aaaaaaaaa bbbbbb n/f eof

第一部分 10 位数字 aaaaaaaaaa 代表该对象的字节偏移，第二部分五位数字 bbbbbb 为其生成号，第三部分 n/f 代表该项是否有效，最后是一个行结束标志 eof。每个空闲项包含下一个无效对象号，所有的空闲项构成了一个链表。而交叉引用表的第一项（对象号为 0，生成号是 65535）永远是空闲项，就成了此无效对象链表的表头，而链表的最后一项将指回对象号 0 的对象。如果交叉引用项在重复使用，其生成号就会不停的加 1，直到计数为 65535 时，此项将变为空闲项。

#### 2.2.4 文件尾

PDF 文件的尾部是解析整个文件的入口，通过逻辑结构解析 PDF 文件，须从尾部着手。通过读取尾部，可以快速定位交叉引用表和特定对象的位置，所以传统的 PDF 文件解析从尾部着手。PDF 文件以标识符%%EOF 结束，前面分别是关键词“startxref”和最后一个交叉引用项的偏移地址，如图 2-7 所示。

```
183  trailer
184  << /Size 12
185  /Root 10 0 R
186  /Info 11 0 R
187  /ID [<3E02A947A1039D533D2BAC80E1AB307E>
188  <3E02A947A1039D533D2BAC80E1AB307E>] >>
189  startxref
190  13778
191  %%EOF
```

图 2-7 文件尾结构图

关键词“startxref”前面是一个字典对象，其由一个关键词“trailer”和紧随其后的由<< >>括起来的一系列键值对组成，字典对象的键值对介绍如表 2-3 所示。

表 2-3 文件尾字典对象键值对

Key	Value
Size	指出有多少条交叉引用项
Root	指出 PDF 文档的根目录的间接对象
Encrypt	PDF 文档的加密，本文针对的都是未加密的
Info	PDF 文档的信息字典对象
ID	文本标识

2.3 逻辑结构

上一节所介绍的文件结构是物理结构，它说明了二进制数据存放的格式。而逻辑结构反应的是逻辑关系，这是解析传统 PDF 文件的基石。PDF 文档具有层次性（树形结构），这些对象都存储在 PDF 文件 Body 部分。首先，其根部是文档目录 Catalog 字典对象，包含对 Pages 对象的描述，而文档的每一页都由一个页面对象表示，页面对象是一个字典对象，包含对页面内容和其它属性引用信息，如注释、图像、文本等。每个页面对象被绑定在一个结构中构成页面树，页面树由文档目录字典中的一个间接对象引用指明。该层次结构中，父亲、孩子、兄弟关系节点都由字典对象中相关项定义，通过引用实现其定义。一个典型的 PDF 逻辑结构示意图如图 2-8 所示。

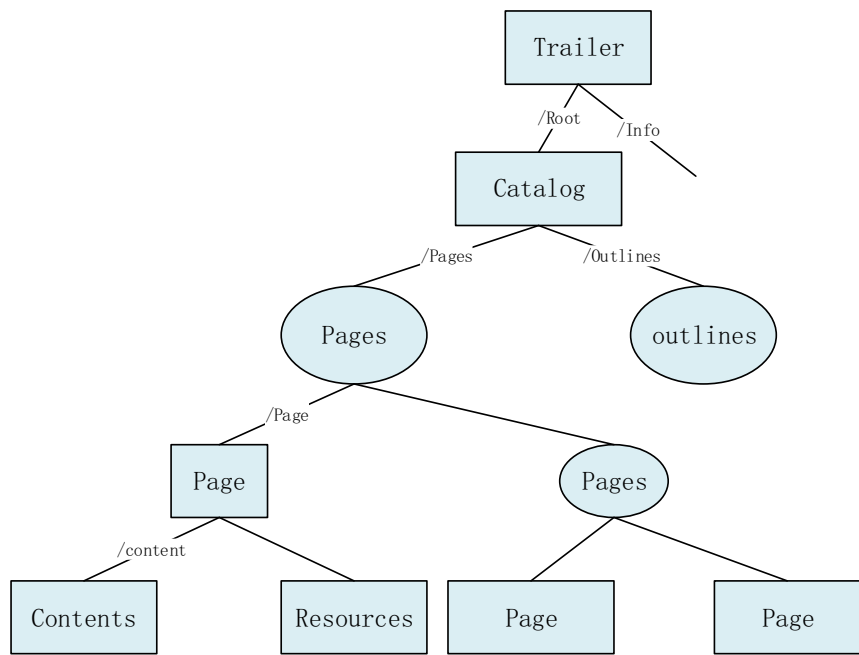


图 2-8 PDF 文件逻辑结构示意图

2.3.1 文档目录

在文档对象层次结构中，目录字典对象作为该结构的根部，可通过尾部信息获取该字典对象的信息，由尾部信息的 Root 项获得，如 /Root 10 0 R，可知 Catalog 对象通过对象号 10 间接引用。文档目录字典中包含了其他间接引用信息，如文本内容、大纲等。如图 2-9 所示，其中 pages 项的值是对象 5。

```
157 10 0 obj <<
158 /Type /Catalog
159 /Pages 5 0 R
160 >> endobj
```

图 2-9 文档目录示意图

此外，目录字典对象有些还包含文档显示的方式，例如大纲、页缩略图是否自动显示等。目录字典的主要属性如表 2-4 所示。

表 2-4 目录字典属性表

Key	Value
Type	其值为 Catalog，表明该对象是目录，属于字典对象类型
Pages	是一个间接引用，是页面树的根
Outlines	是一个间接引用，其值为大纲字典对象
Thread	是一个间接引用，其值为线索字典对象

2.3.2 页树

PDF 文档中的的每一页都由一个页面对象（page）构成，这些页面通过页树对象

(Pages) 间接访问。由于页树定义了文档中页面的先后顺序，因此该结构可以使应用程序花少量的时间、占少量的内存打开 PDF 文档。页面树的中间节点都是一个子页树，而叶子节点都是页面对象。如图 2-10 所示，Kids 就是一个中间节点，是一个子页树，页树节点中各项见表 2-5 所示。

```
152 5 0 obj <<
153 /Type /Pages
154 /Count 1
155 /Kids [2 0 R]
156 >> endobj
```

图 2-10 页树示意图

表 2-5 页树节点项

key	Value
Type	其值为 pages，类型为字典对象
Parent	指向父节点的间接对象引用
Kids	数组对象类型，指出其叶子节点对象引用
Count	整数对象类型，指出其叶子节点的数目，即页面数

2.3.3 页节点

页树的叶子节点是一个页面对象，其类型也是个字典对象，由字典的项来描面页的属性，对解析最重要的是文本内容对象，如图 2-11 所示。页节点属性介绍如表 2-6 所示。

```
12 2 0 obj <<
13 /Type /Page
14 /Contents 3 0 R
15 /Resources 1 0 R
16 /MediaBox [0 0 612 792]
17 /Parent 5 0 R
18 >> endobj
```

图 2-11 页面示意图

表 2-6 页面属性

Key	Value
Type	其值为 page，指明是一个页面对象
Parent	指向父节点的间接对象引用
Resource	描述页面所用的资源，如字体、进程集等
Content	通过引用对象表明的该页面的内容，包含我们所需的文本内容，该项可以为流、流数组

## 2.4 内容流

PDF 文档中的每页都是由一个或多个内容流来进行描述的,即是描述页面显示和其它图形元素的主要手段,依赖于一个相关的资源词典,这两个对象形成一个独立的实体。

内容流是一个 PDF 流对象,其内容由一系列操作数及其参数构成,用来描述页面的图形元素,这些指令以对象形式存放。对内容流进行分析解析前,需对内容流进行解压缩,绝大多数内容流都是被压缩过的。

### 2.4.1 Stream 流的解/压缩方法

以上小节详细介绍了 PDF 文档的格式,在 PDF 的流对象中,一般 stream 与 endstream 之间的字节都是被压缩的,解析并提取 PDF 文件的文本内容需要先解压缩,再对可视化的信息进行处理,利用编码规则来提取文本。

由于存储空间等原因,大部分的 PDF 文档的内容流都是被压缩过的,Filter 作为流规范的可选部分,指示数据中的内容流必须在被使用之前进行解压缩处理,这个指示信息在流字典的 Filter 的项中,如/Filter /FlateDecode,相关的压缩方法如下表 2-7 所示。

表 2-7 常用的压缩方法

Filter	Description
ASCIHexDecode	用 ascii16 进制编码过的数据
ASCII85Decode	用 ascii85 编码过的数据
LZWDecode	用 LZW 自适应压缩过的数据
FlateDecode	用 Zlib/deflate 压缩过的数据

由于 PDF 文件的文本内容流及文本编码、CID<sup>[41]</sup>编码的内容流通常采用的是 FlateDecode 压缩方式,我们将在 PDF 文本内容提取章节详细介绍该压缩方式。

### 2.4.2 解压后的内容流

解码后的内容由一系列操作数和操作符构成,除流对象以外,操作数必须是直接对象,不能为间接对象及引用对象。操作符是指定要执行的操作,比如在页面上绘制一个图形或显示某些文本内容。需注意该操作符区别于名称对象是前面没有“/”,且操作符只有在内容流内才有意义。

一个 PDF 文本对象可以包含要显示的文本字符串内容,其在页面的位置,设置文本状态和其他参数等一系列操作。对于文本流解压缩后的主要内容如下:

- 1) 文本对象:主要提供了文本的相关信息,如显示的文本内容、字体相关信息、位置等格式信息,一个文本对象开始于操作符 BT,结束于操作符 ET。

- 2) 文本字体信息：通过 Tf 操作符设置字体信息。Tf 有 2 个参数，分别代表字体的名称和大小。
- 3) 文本位置信息：操作符 Td/TD 用于设置文本行的位置，Td/TD 也有参数 Tx 和 Ty(分别表示描述当前行的水平、垂直位移)。
- 4) 文本矩阵：描述字体和换行信息。通过文本矩阵操作符 Tm 来设置文本矩阵，即[Sx 0 0 Sy Tx Ty]，其中 Tx 和 Ty 含义如上，参数 Sx 和 Sy 表示字体的宽度和高度。
- 5) 文本内容：Tj/TJ 操作符用于写入文本内容，即我们需要提取的文本内容，其位置在操作符前括号内的文本串。

## 2.5 中文的转码

从解压缩后内容流格式可知，PDF 文本的内容在 TJ/Tj 操作符的前面，英文的文本内容一般在一对圆括号里，中文的文本内容一般在一对尖括号里面；如果寻找到的的是 Tj 参数，那么就提取括号里的内容；如果寻找到的的是 TJ 参数，说明在每对括号的前面还有一些控制信息，这时需要去掉这些数字控制信息，再提取出括号内的内容；如果遇到转义字符“\”标志则不用提取，去掉即可。

但因中文字符通常是由多个字节构成的宽字符，和英文字符数量比较，中文汉字数量很大，因此在 PDF 文件中，为了减少文件大小，PDF 文件为中文汉字使用 CID 编码<sup>[42]</sup>。在 PDF 文件中正文内容的编码不是 Unicode 编码时，为了得到能够显示的 Unicode 编码，需要一个从 CID 编码到 Unicode 编码的转换过程，其转换过程示意如图 2-12 所示。

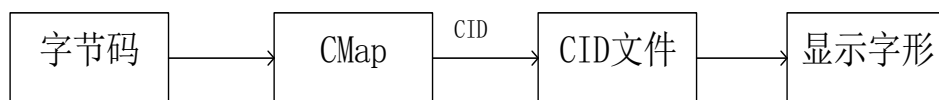


图 2-12 编码转换图

### 2.5.1 Cmap

Cmap<sup>[41]</sup>详细说明了从字节码到 CID 的对应映射关系。一个 Cmap 可以通过以下 2 种方式说明：

- 1) 一个 name 对象标识了一个预定义的 Cmap 文件；
- 2) 一个 stream 对象的内容是一个 Cmap 文件。

预定义的 Cmap 指已经知道字节码到 CIDs 的映射，有预设的 CJK（中国、日本、韩国）的 Cmap；如果字符编码没有预设，那么 PDF 文件必须包含一个 stream 来定义 Cmap，通过解压该流，得到 Cmap 字典信息，相当于字符集的相关信息，这个 Cmap stream 字典包含的主要信息如表 2-8 所示。



表 2-8 Camp stream 流字典信息表

Key	Value
Type	CMap
CMapName	如 90ms-RKSJ-H
Registry	定义字符集的发行者，如 Adobe
Ordering	指定的 Registry 的字符集，如 Japan1
Supplement	字符集的补充，默认为 0，有 CID 加入这个字符集，则依次加 1
WMode	书写模式：0 水平，1 垂直，默认为 0
UseCMap	预定义 cmap 的名字，或是 CMap 流

如图 2-13 所示，图 2-13 给出了一个解压后的 Cmap 文件的二例。

```

22 0 obj
  << /Type /CMap
    /CMapName /90ms-RKSJ-H
    /CIDSystemInfo << /Registry (Adobe)
      /Ordering (Japan1)
      /Supplement 2
    >>
    /WMode 0
    /Length 23 0 R
  >>
  stream
    /CIDInit /ProcSet findresource begin
    12 dict begin
    begincmap
    /CIDSystemInfo
    3 dict dup begin
    /Registry (Adobe) def
    /Ordering (Japan1) def
    /Supplement 2 def
    end def
    /CMapName /90ms-RKSJ-H def
    /CMapVersion 10.001 def
    /CMapType 1 def
    /UIDOffset 950 def
    /XUID [1 10 25343] def
    /WMode 0 def
    4 begincodespacerange
    <00> <80>
    <8140> <9FFC>
    <A0> <DF>
    <E040> <FCFC>
    endcodespacerange
    . beginnotdefrange
    <00> <1F> 231
    endnotdefrange
    100 begincidrange
    <20> <7D> 231
    <7E> <7E> 631
    <8140> <817E> 633
    <8180> <81AC> 696
    <81B8> <81BF> 741
    <81C8> <81CE> 749
    ... Additional ranges...
    <FB40> <FB7E> 8518
    <FB80> <FBFC> 8581
    <FC40> <FC4B> 8706
    endcidrange
    endcmap
    CMapName currentdict /CMap defineresource pop
    end
    end
    %%EndResource
    %%EOF
  endstream

```

图 2-13 Cmap 示例图

Cmap 文件各个部分的作用说明如下：

- 1) begincmap 和 endcmap: 这两个字符串间存放 cmap 的定义；
- 2) usecmap: 合并了来自另一个 Cmap 文件的代码映射，指字符码映射到另外一个 cmap 文件；
- 3) begincodespacerange 和 endcodespacerange: 定义编码的范围；
- 4) usefont: 说明 font 号，其值必须为 0，但是一般不出现；

- 5) `beginbfchar` 和 `endbfchar`, `beginbfrange` 和 `endbfrange`: 输入的字符码映射为相关字体的字符码或字形名字, 如果是 Type 0 字体, 这些将在 `ToUnicode Cmap`<sup>[43]</sup> 呈现, 下一小节会进一步阐述;
- 6) `begincidchar` 和 `endcidchar`, `begincidrange` 和 `endcidrange`: 输入的字符码映射为相关字体的 CID;
- 7) `beginnotdefchar`, `endnotdefchar`, `beginnotdefrange` 和 `endnotdefrange`: 字符码到 CID 映射未定义的字符及区间。

### 2.5.2 字符码转为 unicode 码

一个 PDF 文档阅读器, 需把字符码先转换为 Unicode 码, 才能解析该 PDF, 显示出字形。以下是字符码转换为 unicode 码的几种方法:

- 1) 如果字体字典包含有 `ToUnicode Cmap`, 则用这个 `Cmap` 直接将字符码转换为 Unicode。
- 2) 如果字体为简单字体, 用 `MacRomanEncoding`、`MacExpertEncoding`、`WinAnsiEncoding` 任意一种预定义编码, 或有 `Differences array` 的编码, 其包含 Adobe 标准拉丁字符集或 `Symbol font` 的字符名字:
  - A. 通过 `Differences array` 或字符集直接将字符码转换为字符名字;
  - B. 在 `Adobe Glyph List` 表找到字符名字获得相应的 Unicode 值。
- 3) 如果字体为复杂字体, 用预定义 `Cmap`, 或使用 the `Adobe-GB1`, `Adobe-CNS1` 等字符集:
  - A. 通过个字体的 `Cmap` 把字符码转换为 CID;
  - B. 通过这个 `Cmap` 使用的字符集从 `CIDSystemInfo` 字典获得其 `registry`、`ordering`;
  - C. 通过 B 步骤获得的信息, 构建二次 `CMap name`, 如 `registry-ordering-UCS2`;
  - D. 通过 C 步骤获得这个 `Cmap name` 获得 `CMap`(可从 ASN 网站上获得);
  - E. 从 A、D 步骤获得的 CID 及 `Cmap`, 把 CID 转换为 Unicode 值。

### 2.5.3 ToUnicode Cmap 的转换

以上小节介绍的编码转换关系, 大多数为标准类型的 PDF 文件的编码转换, 而对于某些非标准的 PDF 文件, 采用更简单的编码转换方法。为了从 PDF 文档中快速显示内容, 采用了在文件中内嵌编码转换表, 即 `ToUnicode Cmap` 文件, 避免了读取外部 `Cmap` 文件, 提高了效率。

为确保可移植性, 与 `Encoding` 属性后给出的 `Cmap` 不同, `ToUnicode CMap` 不能通过 `name` 对象预设一个 `ToUnicode CMap` 对象, 而是内嵌 `ToUnicode` 中的 `Cmap` 映射,

需通过 PDF 文件中的文字内容，独立对应生成的编码转换，直截了当地显示转换关系，版本为 1.3 和 1.4 的必须使用 ToUnicode 映射。

同时 ToUnicode 映射文件遵循 Cmap 文件语法，“ToUnicode”映射文件的名称由三个部分组成：/Registry string, /Ordering string, /Supplement integer, 如 Adobe-UCS-0。从 CMapType 2 我们可以看出该映射文件已经完成 CID 转换为 unicode 码。从映射方式我们可以看出单一映射和区间映射（即 1 对 1，多对多），解压后的 ToUnicode 文件示例如图 2-14 所示，具体对 ToUnicode Cmap 文件的解释说明如下。

```
78683 150 dict begin
78684 begincmap
78685 /CIDSystemInfo
78686 << /Registry (Adobe)
78687 /Ordering (UCS)
78688 /Supplement 0
78689 >> def
78690 /CMapName /Adobe-Identity-UCS def
78691 /CMapType 2 def
78692 1 begincodespacerange
78693 <0000> <FFFF>
78694 endcodespacerange
78695 2 beginbfchar
78696 <0066> <00B1>
78697 <00FC> <2014>
78698 endbfchar
78699 3 beginbfrange
78700 <0100> <0101> <201C>
78701 <01C3> <01C4> <3001>
78702 <02C4> <02C5> <FF08>
78703 endbfrange
```

图 2-14 ToUnicode 文件图

ToUnicode 中详细的编码转换规则（映射）有如下两种：

1) 单一对应（一对一）

单一对应规则：操作者为“beginbfchar”和“endbfchar”，在操作者之间的 16 进制数是一一对应，相当于字符码直接对应 Unicode 码，如“<0066> <00B1>”表示为 PDF 文本内容流中出现的是“0066”，其对应的 Unicode 码是“00B1”，对应字形为“一”。

2) 区间对应

区间对应规则：操作者为“beginbfrange”和“endbfrange”，在操作者之间的 3 组 16 进制，依次对应，其表示为：“开始的字符码”“结束的字符码”“Unicode 码”，如：“<01C3> <01C4> <3001>”表示“01C3”对应 Unicode 码“3001”，“01C4”对应 Unicode 码“3002”。

---

## 2.6 本章小结

本章对 PDF 文件的格式进行了深入的研究，分别从其对象、逻辑结构、物理结构、编码及转换方式进行了研究；尤其是逻辑结构及编码转换方式为后章的文件解析奠定了基础。

## 第 3 章 PDF 文档解析及脱敏处理

在第二章，我们主要从 PDF 的语法结构入手，详细介绍了 PDF 文档的文件格式，深入研究 PDF 文件的格式、逻辑结构、压缩方式、编码方式等。结合第二章的内容，这一章我们将深入研究 PDF 文件的解析方法，即根据其逻辑结构来进行文本内容的解析，之后结合 PDF 文件结构，从 Stream 流入手介绍 PDF 文件脱敏处理流程。

### 3.1 PDF 文档内容解析

正如 2.3 节所介绍的 PDF 文件的逻辑结构，整个 PDF 文件流的内容很多，每个部分都是由各种基本数据类型的对象组成，因此根据其逻辑结构可以定位文本内容。PDF 文件的解析入口是文件尾 Trailer，根据尾部的 Trailer 提供的交叉应用表的信息和根对象号，然后根据交叉引用表所提供的各个对象的相对位置、以及根对象下面的层次结构就可以读取一个 PDF 文件所有内容。如图 3-1 所示，PDF 文档的完整解析步骤如下：

#### 1) 查找文件尾 trailer

一个 PDF 文件的尾部包含了关键信息，其可以使我们快速定位到交叉引用表和某些特定对象，所以一般解析完整的 PDF 文件都是从尾部开始，指针定位到文件尾，文件结束标志%%EOF。

#### 2) 定位 PDF 的根对象 Root

从文件尾 trailer 中找到其属性标签为/Root，即 PDF 层次结构的根节点，获得其间接对象号，再通过交叉引用表对该对象定位操作。

#### 3) 查找 Catalog 目录对象

通过 2) 获得 Root 间接对象，其类型 type 为 Catalog，属性标签为/Catalog（从 1.5 版开始，trailer 和交叉引用表合并在一个对象里，通过关键字“startxref”后的字节偏移，找到这个对象，并找到 root 根对象）。

#### 4) 定位页树节点对象 Pages

通过 Catalog 根节点找到属性标签/Pages，获得其间接对象号，结合交叉引用表定位到 Pages 对象。

#### 5) 查找页对象 Page

通过 Pages 子节点找到属性标签为/page 的页对象，并取得属性标签 Kids 其后的间接对象号，这个对象号可能是一个，也可能是多个数组，视页面的情况而定。

#### 6) 访问页对象 Page 中的内容

定位页对象后，这时类型 type 为 Page，寻找属性标签/Contents，如果找不到属性标签 Contents，则说明此页内容为空，不予处理；如果存在属性标签 Contents，则进行后续处理。

## 7) 记录所有 Content 对象号

将通过寻找标签 Content 的所有间接对象号记录于内容对象号数组，以便后续进行解析。

## 8) 定位内容对象号数组中的对象

根据数组中存储的对象号，结合交叉引用表，根据相对偏移位置直接定位到该对象。

## 9) 提取部分 stream 流内容

依次提取以上对象的流内容，读取 stream 和 endstream 中的内容，但大多数流内容都是经过压缩后的；如果对其直接读取出来，看到的是乱码，因此需要先通过该对象获取属性标签 Filter 后的压缩方法，并将 stream 与 endstream 之间的所有内容存入数组中。

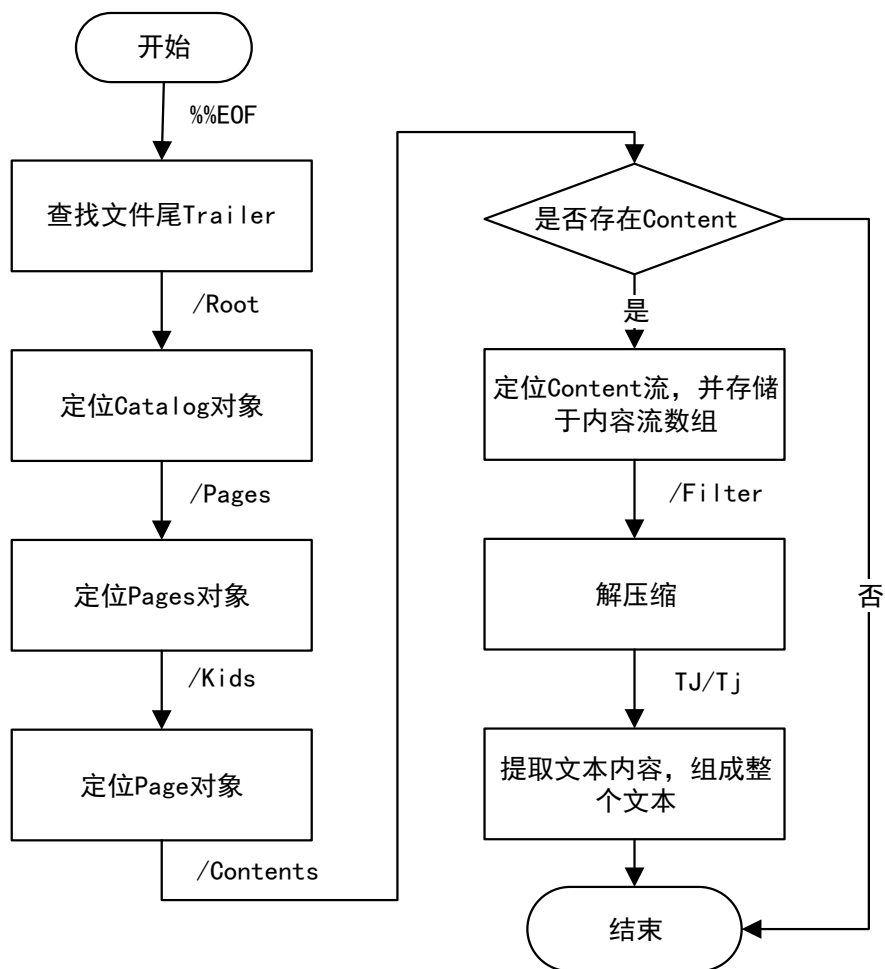


图 3-1 PDF 解析流程

## 10) 解压缩

文本内容流通常采用的压缩算法为 FlateDecode，可以通过调用 zlib 库对数组中的对象的内容流进行解码。

## 11) 提取文本内容

对解压缩后的数据，文本内容放在标识符 BT 和 ET 之间，标识符之间有各种位置信息、文本信息、字体等信息，但是真实文本内容在操作符 TJ/Tj 前面的圆括号或尖括号后，因此考虑将 TJ/Tj 前面的内容存到一个数组，再将所有包含内容对象的信息进行拼接，组装成整个 PDF 文档的文本内容。

以下我们以图 3-2 的部分字节流为例具体讲述 PDF 文件的解析流程。

```

1  %PDF-1.5
2  %µµµµ
3  1 0 obj
4  <</Type/Catalog/Pages 2 0 R/Lang(zh-CN) /StructTreeRoot 34 0 R>>
5  endobj
6  2 0 obj
7  <</Type/Pages/Count 6/Kids[ 3 0 R 23 0 R 25 0 R 27 0 R 29 0 R 31 0 R ] >>
8  endobj
9  3 0 obj
10 <</Type/Page/Parent 2 0 R
11 /Resources<</Font<</F1 5 0 R/F2 8 0 R/F3 13 0 R/F4 15 0 R/F5 20 0 R>>
12 /ExtGState<</GS7 7 0 R>>/ProcSet[/PDF/Text/ImageB/ImageC/ImageI] >>
13 /MediaBox[ 0 0 595.32 841.92]
14 /Group<</Type/Group/S/Transparency/CS/DeviceRGB>>/Tabs/S/StructParents 0>>
15 endobj
16 4 0 obj
17 <</Filter/FlateDecode/Length 1920>>
18 stream
19 xœZYESC
20 ~_`ÿC?J[SOHÔ[-b]À`>
21 BEL1`ÃESCäAðf ÈŠRS,ÅÎSTXAp}HÖ±Ý³%5³+é™i.PäGrî+?RSçüúääcw:Ý
22 US>pèÓ?»÷wã·ÇCoçýr÷ðçΠ²ûñÃç/_?<~ûððb%ESCç@ESCUSnoiVèb÷ðëí
23 tŠp;øWSUB;VT½×YÃoðÈw?ûiônoT÷™ÿ÷ÝiÎû7YÛ_²tçpp,tÆi·7:ÚRSç±±GCUS=b
24 #ùibÄ -QÊéÛI. !×ESCk•xkjÉRSX...üŠ<ëÖ*FS ENQİsÜë`Ty~
25 ETB™STX-zİİÔBùnvs
26 «xÚ‡|^ăkq‘ëýkdĐêu
27 endstream
28 endobj

6552 trailer
6553 <</Size 216
6554 /Root 1 0 R/Info 33 0 R
6555 /ID[<DD4264A0AE02E148AD5E0B44038E6042><DD4264A0AE02E148AD5E0B44038E6042>] >>
6556 startxref
6557 217578
6558 %%EOF

```

图 3-2 部分字节流图

针对图 3-2 所示的字节流，PDF 文件解析流程如图 3-3 所示。

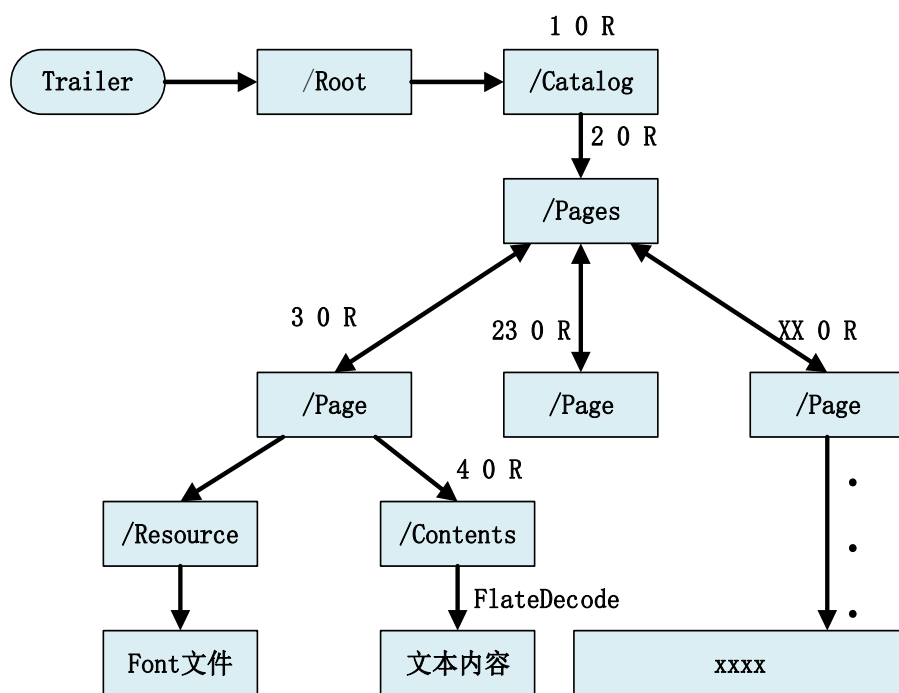


图 3-3 PDF 示例解析流程

### 3.2 基于 Stream 流的 PDF 文档解析

沿着 3.1 节所介绍的步骤可以解析出 PDF 文件内容，但由于 PDF 文档由一系列对象构成，对象种类也较多，处理比较繁琐，并且有些版本的尾部信息压缩在某个流里，还得提前解压流，才能处理尾部信息，这使得传统的 PDF 文件解析方法复杂度较高。为了简单快速对 PDF 文件内容进行解析，实现高效的敏感信息的脱敏处理，我们采取直接基于 Stream 流的 PDF 文档内容解析处理。针对 PDF 流解析，以下从 PDF 文件识别、分段解压及封装、Stream 流信息的分类、萃取 Cmap、正文文本及注释的提取介绍 PDF 文档的解析处理方法。

### 3.2.1 PDF 文件识别

PDF 文件识别的主要任务是如何判断一个文件是否为 PDF 文件。在抓包分析判断截获到的文件是否为 pdf 文件时,为了避免某些非 PDF 文件将后缀改为.pdf,除了文件的后缀名,我们还需要做第二次识别,即对该文件的字节流进行分析。

由 2.2 节 PDF 文件的物理结构可知, PDF 文件头有 PDF 文件的标识, 即 PDF-1.x, 证明其为 PDF 文件及其版本号。若没有识别到 PDF 文件的标识, 则不对其处理, 并返回-1。若该文件为 PDF 文件, 则获取其版本号, 并返回 0, 继续后面操作。

### 3.2.2 分段解压及封装

通过 2.4 小节,我们知道 PDF 文件的 stream 流内容基本是不可见的,而在解析 PDF 文件时所需的文本内容及编码方式等都在 stream 流里。由于 PDF 文件中的 stream 流信



息通常都是采用 FlateDecode 方式压缩, 因此需要进行解压处理。在具体应用中, 我们采取调用 Zlib 库的方法来完成解压, 所用到的工具函数主要是 `inflate_buf`。

在对 PDF 文件解析并完成 stream 流中的敏感内容进行内容脱敏处理后, 需要在不改动 PDF 文件格式的情况下, 将脱敏的文本流的内容原封不动的压缩回去, 即封装过程, 可采用调用 Zlib 库来实现 FlateDecode 方式压缩, 所用到的工具函数主要是 `deflate_buf`。

调用 Zlib 库对 stream 流的解压过程与一般解压过程一样, 都是依次调用接口函数 `inflateInit()`, `inflate()`, `inflateEnd()`; 解压缩做脱敏处理后, 还得封装压缩回去, 即压缩过程: 依次调用接口函数 `deflateInit()`, `deflate()`, `deflateEnd()`。有关解压缩及压缩过程用到信息的相关数据结构及接口函数说明如下:

### 1. z\_stream 结构

在调用这些接口函数前, 先定义 `z_stream` 结构体, 存放解压缩及压缩的相关数据信息如下:

`next_in`: 存放准备压缩数据的首地址;

`avail_in`: 存放准备压缩数据的长度信息;

`zalloc` 和 `zfree`: 分别用于分配和释放内部状态, 调用初始化函数前, 其值需初始化为 `Z_NULL`;

`opaque`: 私有数据对象, 传递给 `zalloc` 和 `zfree` 中的第一个参数, 调用初始化函数前, 其值需初始化为 `Z_NULL`;

`next_out`: 存放被压缩数据后的首地址;

`avail_out`: 存放被压缩数据后的最大长度。

### 2. 函数调用

函数初始化: `inflateInit` 函数和 `deflateInit` 函数, 每次解压缩、压缩之前, 需对 `z_stream` 结构对象进行初始化, `inflateInit` 函数只有一个参数 `z_stream`, 而 `deflateInit` 函数有 `z_stream` 和压缩等级 `level` 两个参数, 压缩等级 `level` 采用默认压缩等级 `Z_DEFAULT_COMPRESSION`。

`inflate` 函数: 在 `inflateInit` 初始化后, 对 `z_stream` 中的 `next_in`、`avail_in`、`next_out`、`avail_out` 赋值, 设置刷新参数方式为 `Z_NO_FLUSH`, `inflate` 的两个参数为 `z_stream` 和 `Z_NO_FLUSH`。

`deflate` 函数: 在 `deflateInit` 初始化后, 对 `z_stream` 中的 `next_in`、`avail_in`、`next_out`、`avail_out` 赋值, 设置刷新参数方式 `flush`, 本文所有要进行编码的数据都已经在 `next_in` 中, 其值一般设置为 `Z_FINISH`, `deflate` 的两个参数为 `z_stream` 和 `flush`。设置完成后调用 `deflate` 函数即可, 如果有错误出现, 调用 `deflateEnd` 函数。

### 3.2.3 Stream 流信息的分类

PDF 文件是由一系列的对象组成的,通过对 3.1 节传统 PDF 文件的解析分析可知,我们所需要的 PDF 文本的内容(除正文内容外,还包含部分注释)、字体信息(Cmap 映射)都是存储在 stream 流里面的,因此我们可以通过直接定位 stream 流来快速获取这些信息。

利用单模式匹配算法寻找关键字“stream”和“endstream”,分别代表流的开始与结束,其中被压缩的内容可能是我们所需处理的文本内容。通过调用工具函数 inf\_buf,解压其间的的内容,解压成功,才进行后续处理。由于 stream 流的内容在解压之前,无法知道其属于文本内容流还是 Cmap 映射流,因此需要对这些流进行分类处理。

#### 1. 文本内容流

正文文本内容流的标识是“BT”,代表文本的内容的开始;注释文本的标识是“Contents”,注释存在的位置分为两种情况,一是和正文文本内容一样,直接被压缩在 stream 流里,另一种是在正文文件结束符“%%EOF”后,存在一段注释流。

通过快速单模式匹配寻找关键字“BT”和“Contents”,若找到返回 1,分别设置压缩的类型 inftype 来代表文本被压缩和注释被压缩(1 代表文本压缩,2 代表注释压缩),并用内容文本对象记录每个文本内容对象流的相对位置,以便后续解析操作。

#### 2. Cmap 映射流

如果解压出的流的内容并不是正文文本内容或注释的内容,则考虑为 Cmap 映射流的内容。Cmap 映射流的标识是“begincmap”,亦是快速单模式匹配查找该标识,若查找成功,则添加为 Cmap 对象,进行后续 Cmap 映射流的处理。

### 3.2.4 萃取 Cmap 流

如 2.6 节所述,Cmap 的出现可能有多种形式,如预设的 Cmap 文件,PDF 文件 stream 流内嵌的 Cmap 文件。PDF 文件的文本内容编码方式可能存在多种情况,如 CID 编码(Cmap 文件映射)、GBK 编码、unicode 编码等,因此需要考虑对每种情况进行 Cmap 的提取,以便能对 PDF 文件进行准确解析。Cmap 文件的提取过程如下:

#### 1. 提取 stream 流的 Cmap 文件

如上小节所述,在对 stream 流进行信息统计时,当判断到该流是 Camp 映射流时,便进行 Cmap 文件的提取和存储,Cmap 映射文件一般以两种格式存在,单一对应和区间对应。对应的可以分两种情况处理:一是利用快速单模式匹配算法对这段 Cmap 文件流查找关键字“beginbfchar”,若找到代表是单一对应,则直接按一对一(Key—Value)方式提取其“beginbfchar”和“endbfchar”之间的字符码及对应字形码;二是利用快速单模式匹配算法对这段 Cmap 文件流查找关键字“beginbfrange”,找到代表这部分是区间对应,提取“beginbfrange”和“endbfrange”间的 Cmap 信息,先计算该区间的长

度，分别对字符码和其对应的字形码依次累加并存储，再一对一地提取这区间的所有 Cmap 映射。

## 2. 提取预设 Cmap 文件

某些 PDF 文件预设了 Cmap 映射文件，为避免浪费内存空间大小，我们无需每次都读入，通过 CIDSystemInfo 字典信息，当我们检测到预设的 Cmap 文件的名称，如 Adobe-GB-UCS2，这时我们才读取该 Cmap 文件的内容，提取字符码和字形码之间的对应关系，提取方法和提取 stream 流的 Cmap 文件一样，这里不再赘述。

由于 GBK 和 Unicode 码都有统一的标准，我们本文研究的目的在于把敏感信息脱敏掉，无需再执行这一过程中把这些字形显示出来，我们只需知道其字符码。所以为减小内存空间的消耗，在对敏感信息处理的时候，再利用 GBK 和 Unicode 码。

## 3.2.5 正文及注释的提取

在分析 stream 流时，我们已经确定了文本内容流在内存中的位置，即正文文本内容和注释内容流的位置。通过文本内容对象向量，确定每个流的相对位置，并获得文本内容流和注释流的标识信息 intype，进而分别对这两种类型的流先进行解压缩，再进行文本的提取。

### 1. 正文文本内容流

正文文本内容以标识符“BT”开始，正文文本内容就在标识符“BT”和“ET”之间，前一章介绍 Stream 流信息时，我们知道在 BT 和 ET 之间除了正文文本内容之外，还有其他控制信息，如字体类型及大小 Tf、文本矩阵 Tm 等信息，但是我们所需要的文本内容都在操作符 TJ/Tj 前面的括号里；由于本文主要目的不在于 PDF 提取与显示，而在于对 PDF 文本内容中敏感信息的脱敏处理，因此这里我们只需要考虑文本的具体内容，其他控制信息无需考虑。

全英文的 PDF 文件，其文本内容在 Tj 前面配对的一对圆括号“()”内，此时英文的每个单词间是连续的；或是在 TJ 前面的方括号“[]”的数组里，在方括号内，又有调节对齐等位置的控制信息，但是真实的文本内容还是在配对圆括号“(”和“)”里；因此不管操作符是 TJ 还是 Tj，忽略圆括号之间的控制信息，采取直接提取“()”的内容，如：(Hello PDF) Tj, [(H)80(ell)40(o)-350(wo)45(rld)]TJ，只需把圆括号里的内容提取出来并存放，以便后续进行敏感词的脱敏。当然，全英文的 PDF 文件文本内容也会存在像中文 PDF 文件的格式，我们后面再详述。

中文或中英混合的 PDF 文件，其文本内容也是在 BT 和 ET 之间，和英文的 PDF 文件一样，其间也有控制信息。前面讲述文本内容编码种类复杂多变，但是无论是 unicode 编码、GBK 编码、还是 Cmap 映射编码，大多数中文 PDF 文件的文本的内容在尖括号“<”和“>”中，其间的文本内容字节码以十六进制的方式表示，和英文 PDF

文件一样，操作符也存在 TJ 和 Tj，如：<2B09086F3E2003EE05B5164302B8>Tj，[<2B09>90<086F3E20>-300<03EE05B51643>100<02B8>]TJ，和某些全英文文件不同，这些十六进制需要通过 Cmap 转码映射文件、GBK 编码、Unicode 编码进行映射才能得到真正的文本内容（显示的字形）；不过有少部分的 GBK 编码在 PDF 文件字节流是以 8 进制呈现，而非 16 进制；基于需求，暂时无需考虑在 PDF 文件流呈现的方式，仅提取成对尖括号和成对圆括号里的字符码内容并存放。

## 2. 文本注释内容的提取

针对 PDF 文件，如果在 acrobat 阅读器中添加了注释，则 PDF 文件会将包含有新增注释文本的数据存放在文件最后或在某个 stream 对象中。若存放“%%EOF”于之后，注释部分的文本内容会存放在某个 pdfobj 对象的“Contents( ... )”中，或者和正文文本内容流相似，压缩在某个 stream 流里面。其中英文及数字存储方式为 ASCII 码的形式且内容是明文，而中文存储为 Unicode 码的形式且内容为被压缩后的。

通过分析截获到的 stream 流的信息，定位到文本注释流，利用工具函数 in\_buf 对 stream 流进行解压缩操作，再利用快速单模式匹配算法，定位到 Contents，提取其后成对圆括号的内容，即文本内容；对于放于尾部的注释，定位到两个“%%EOF”标识符之间的内容，解压缩操作后，开始匹配 Contents，成功匹配后提取其后成对圆括号的内容，即可提取到文本注释内容。

## 3.3 脱敏操作

传统脱敏算法主要有<sup>[44]</sup>：遮挡，即利用“\*”符号替代、空值插入、数值变换、混洗、替代、加密等 6 种方法，在本文我们采取的是遮挡法和空值插入。

对 PDF 文件做解析处理后，便可以进行敏感信息的处理。由于 PDF 文档由一些称为“对象”的模块组成，这些对象可无序地出现在 PDF 文件字节流里，我们对每个 stream 流提取的文本内容可能也是无序的，不过本文目的不在于用文本编辑器来阅读 PDF 文档，因此大可不必关心该顺序，只需判断该文本内容中是否存在敏感信息，即敏感目标的确认。由于 PDF 文件文本内容编码格式复杂，因此考虑不同编码方式的敏感词，以便准确脱敏。

### 1. 获取敏感词存在的不同形式

先将敏感词以 Unicode 码表示，分别获取不同编码的表现形式；再获取在文本中可能出现的形式，如下：

一是通过内嵌 Cmap 文件和预设 CMap 文件，利用快速单模式匹配将敏感词的 unicode 形式匹配其 map 的 Value 值，匹配成功则把其 Key 值存放在敏感词向量的第 5 个位置；

二是将敏感词的 Unicode 表现形式转换为 GBK 编码，以其 GBK 编码形式将敏感

词存放在敏感词向量的第 4 个位置；

三是将敏感词的 Unicode 表现的形式转换为 GBK 编码，再以其 GBK 编码的 8 进制表示形式（\ddd）将敏感词存放在敏感词向量的第 3 个位置；

四是将敏感词的 Unicode 码的形式放在敏感词向量的第 2 个位置；

五是敏感词（原字符串）直接放在敏感词向量的第 1 个位置。

## 2. 敏感词的匹配及脱敏

获取敏感词存在的可能形式后，我们需要对已提取的文本进行敏感词的匹配，即敏感目标的确认，并进行相应的脱敏处理。

在进行敏感词匹配时，基于 PDF 文档编码格式的特点，我们结合经典的单模式匹配算法 BM 和 QS 算法，提出了一种改进后的高效匹配算法，具体细节我们将在论文第四章中详细介绍。

脱敏处理时需注意脱敏后效果，由于不能改变整个 PDF 文件的格式，针对 PDF 文件文本内容的编码格式，对不同编码方式的脱敏处理不同，如：以 Cmap 文件的编码、GBK 形式的敏感词脱敏由空白插入，其他编码形式用“\*\*\*\*”遮挡。

经脱敏处理后，正文和注释文本内容的解压后字节流分别如图 3-4 和 3-5 所示。

```
17 文本内容：电力行业信息化年会论文集
18 原解压字节流
19 [ <2B09>9.5<086F3E2003EE>7.5<05B5164308EA144804EE418E1B5B>8<4C9A>]TJ
20 脱敏词：电力（在 PDF 文件中编码为 2B09086F）
21 脱敏后的解压后的字节流
22 [ <0000>9.5<00003E2003EE>7.5<05B5164308EA144804EE418E1B5B>8<4C9A>]TJ
23 脱敏后的文本内容： 行业信息化年会论文集
```

图 3-4 脱敏后文本内容字节流

```
1 注释内容：为了身体健康，我们要勤练法轮功，努力提高自身修养。
2 原解压字节流
3 ffffffff4e3a4e868eab4f5350655eb7ff0c62114eec898152e47ec36cd58f6e529f002
4 脱敏词：法轮功6cd58f6e529f
5 脱敏后的解压后字节流
6 ffffffff4e3a4e868eab4f5350655eb7ff0c62114eec898152e47ec31f1f1f1f1f002
7 脱敏后的注释：为了身体健康，我们要勤练???.
```

图 3-5 脱敏后注释内容字节流

## 3.4 PDF 文件脱敏流程及实现

### 3.4.1 PDF 文件整体脱敏流程

PDF 文件解析脱敏模块首先将 PDF 文件读入缓冲区，对 PDF 文件进行文件识别及版本号的获取，再同时遍历全部 stream 对象，统计 stream 流的信息，并解压提取文件中的 Cmap 表，将其存入该模块的 map 数据结构中，为之后查表作准备；再把敏感词

的 Unicode 码, 通过 map 反向查表的方式来得到对应敏感词的字符码组合以及敏感词的其他编码组合。准备工作完成后, 定位文本内容流 stream 对象和注释流, 对每个 stream 对象存在的文本内容和 Contents 后的内容进行解压提取, 按照之前求得敏感词组合进行检索并脱敏, 遍历结束后, 脱敏亦完成, 将处理后的缓冲区内容写入新的路径。

基于 Stream 流的 PDF 文件解析及脱敏的实现整体流程图 3-6 如下:

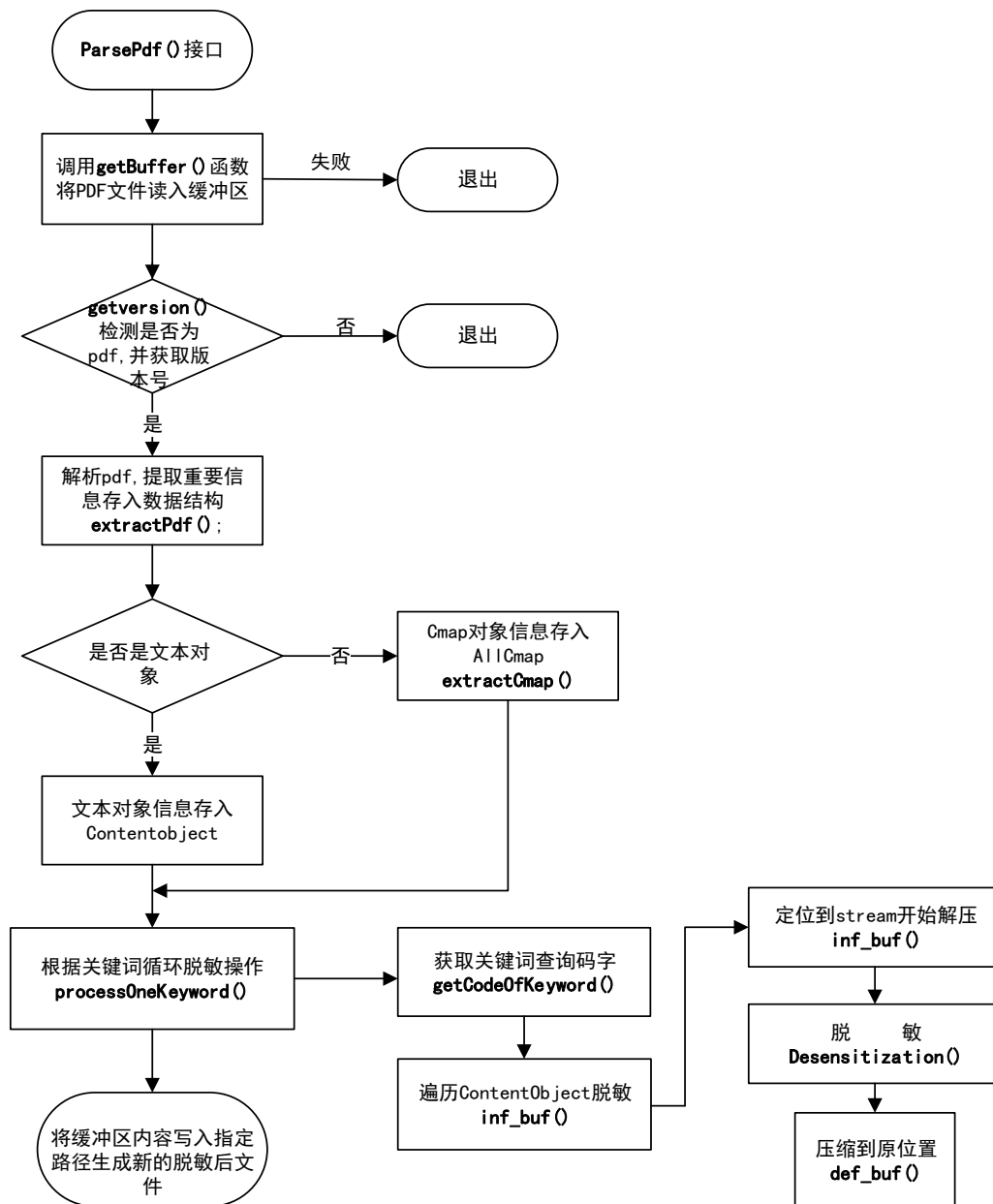


图 3-6 基于 Stream 流的 PDF 脱敏流程图

正文文本敏感词脱敏细节流程图 3-7 如下:

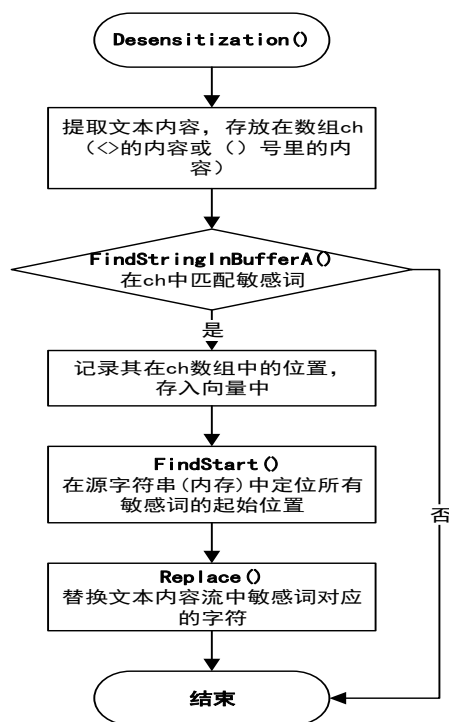


图 3-7 正文脱敏流程图

存放于 `stream` 流的注释文本内容脱敏流程和正文文本内容类似，在文件尾的注释敏感词匹流程图 3-8 如下：

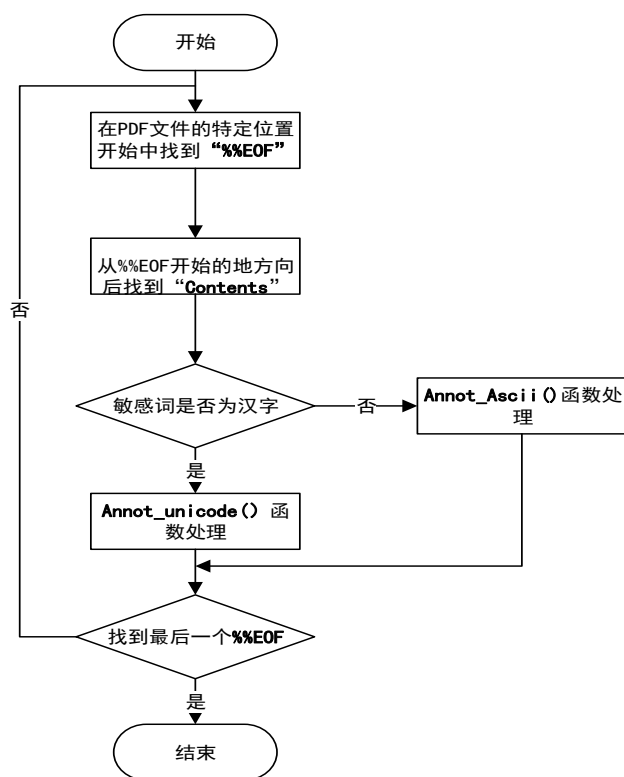


图 3-8 注释脱敏流程图

### 3.4.2 主要类和函数功能实现

#### 1. StreamParser 类

**ParsePdf()**：是主要的接口函数，输入参数：需要脱敏 PDF 文件路径，脱敏后文件存储路径，敏感词集合。返回值：**int** 型。

**getBuffer**：根据路径将 PDF 文件读入缓冲区。

**getversion**：根据读入的字节流，判断是否为 PDF 文件，是返回 0，否则返回-1。

**Check\_FindStringInbuffer()**：用于判断 Stream 流的类型，参数分别为 Stream 流，流类型关键字，流的大小。

**getUnicodeWord**：将敏感词转换为 Unicde 字符串。

**extractPdf**：通过解压判断并统计 Stream 流信息（正文文本流、注释流、Cmap 流）。

**extractCmap**：把该内存区域里的 Cmap 内容存入 Allmap 表。

**processOneKeyword**:针对一个关键词进行全部的脱敏操作。

提取 Stream 流的信息主要代码片如图 3-9。

```
void StreamParser::extractPdf(vector<BTOBJECTPOS> &ContentObject) {
    this->annot_inf_flag = 1; // 说明注释没有被压缩在流中
    定义长度、位置信息等变量
    while (1) {
//匹配Stream流
        streamstart =QS_pdf(buffer, "stream", filelen);
        streamend =QS_pdf(buffer, "endstream", filelen);
        .....
//解压Stream流
        int ret = inf_buf(buffer + streamstart, streamend - streamstart, &output, &output_len)
        if (ret == Z_OK) {
//识别正文文本流
            int rst = Check_FindStringInBuffer(output, "BT", output_len);
//识别注释文本流
            int ast = Check_FindStringInBuffer(output, "Contents(", output_len);
            } ..... //确认注释流是否被压缩
//识别Cmap流
            else int rst = Check_FindStringInBuffer(output, "begincmap", output_len);
            .....
        }
    }
```

图 3-9 stream 流信息主要代码片

#### 2. Cmap 类

**getUCS2**：获得 UCS2 文件中的 cmap 内容，存入 map 表中，有些 PDF 文件会出现缺失 cmap 表，可视情况调用该函数。

提取不同形式 Cmap 方式实现的类似，以下为提取内嵌 Cmap 表的主要代码片，如图 3-10。



```

void StreamParser::extractCmap(char* filebuffer, int filelen, pdfmap* pdfmap) {
    ..... // 定义变量
//匹配cmap表的关键词
    int i1 = QS_pdf(filebuffer, "beginbfchar", filelen);
    int i2 = QS_pdf(filebuffer, "beginbfrange", filelen);
    if (i2 == -1 || (i1>0 && i1 < i2)) {.....//判断是否为单一对应 }
    else if (i1 == -1 || (i2>0 && i2 < i1)) {...判断是否为区间对应 }
    while(pend) { ..... // 定义变量
//提取单一对应的Cmap
        for (i = 1; i < 5; i++) {
            pbegin++; c = *pbegin;
            if(c=='>') break;
            s1 += c; // key值 (字符码) }
            .....
            while (*++pbegin != '>') {
                c = *pbegin;
                s2 += c; //value值 (unicode/GBK) }
                .....
//提取区间对应的Cmap
                for (i = 1; i < 5; i++) {
                    c = *(pbegin + i);
                    s3 += c; }
                    a4 = a3;.....
                    int len = a2 - a1 + 1; //计算区间范围长度
                    .....
                    for (i = 0; i < len; i++) {
                        a4 = a3 + i; //value值
                        a5 = a1 + i; //key键
                        ..... }
//添加到cmap
                pdfmap->cmap.insert(pair<string, string>(s5, s4));
                ..... }
}

```

图 3-10 萃取 Cmap 表代码片

### 3. 工具函数

inf\_buf: 解压函数。

def\_buf: 压缩函数，压缩等级默认用 Z\_DEFAULT\_COMPRESSION。

解压函数实现代码如图 3-11。

```

int inf_buf(char *source, int source_len, char **dest, int *dest_len)
{
    ..... //定义变量并初始化, 分配状态
    ret = inflateInit(&strm);
    .....
    strm.next_in = (Bytef *)source;
    strm.avail_in = source_len;
    do { total_buf += CHUNK;
        buf = (char*)realloc(buf, total_buf*sizeof(uInt));
/*由于buf在做完realloc后, 指针可能改变, 所以temp_buf每次赋值需借助buf的位置*/
        temp_buf = buf + total_buf - CHUNK;
        strm.avail_out = CHUNK;
        strm.next_out = (Bytef *)temp_buf;
        memset(temp_buf, 0, CHUNK*sizeof(uInt)); //初始化
        ret = inflate(&strm, Z_NO_FLUSH);
        .....
        (void)inflateEnd(&strm);
        return ret;}
    have = CHUNK - strm.avail_out; //如果have不是CHUNK, 则说明已经解压完
    temp_buf += have;
    } while (strm.avail_out == 0); //如果avail_out不等于0, 说明所有输入
数据都被解压了; 如果等于0, 则有可能是输出不够了
    total_buf -= strm.avail_out;
    /* done when inflate() says it's done */
    //} while (ret != Z_STREAM_END); //猜测在每一次deflateinit与deflateEnd后,
都会在压缩的数据末尾留下Z_STREAM_END标志
    /* clean up and return */
    (void)inflateEnd(&strm);
    .....
}

```

图 3-11 解压函数代码实现

压缩函数实现代码片如图 3-12。

```
int def_buf(char *source, int source_len, char *dest, int *dest_len, int level)
{
    ..... //定义变量并初始化, 分配状态
    ret = deflateInit(&strm, level);
    if (ret != Z_OK)
        return ret;
    strm.avail_in = source_len;
    strm.next_in = (Bytef*)source;
    strm.avail_out = *dest_len;
    //正常情况下, 数据压缩后一定比原来小, 但也有可能destlen大于source_len
    flush = Z_FINISH; //因为所有要进行编码的数据都已经在source中
    strm.next_out = (Bytef*)dest;
    ret = deflate(&strm, flush); /* no bad return value */
    assert(ret != Z_STREAM_ERROR); /* state not clobbered */
    *dest_len = *dest_len - strm.avail_out;
    assert(strm.avail_in == 0); /* all input will be used */
    assert(strm.avail_out >= 0);
    /*压缩后应该要比压缩前的数据少, 但在simple1中发现一个stream解压后比解压前数据短*/
    (void)deflateEnd(&strm); //对应于deflateinit
    return Z_OK;
}
```

图 3-12 压缩函数实现代码片

#### 4. 脱敏函数

**Desensitization 函数:** 对 output 内容进行脱敏。由于文本内容在 BT 中的括号之中, 所以先将括号中的内容提取到新建的字符数组 ch[]之中, 然后调用函数进行关键词的查找定位, 将位置信息存入到 offset[]数组, 并定位到 output 之中。然后调用 Replace 函数进行脱敏处理。

**CheckAnnots 函数:** 该函数提取存在集合中的每个关键词, 以 string 的形式传入注释脱敏函数进行处理。

**Chenkannot 函数:** 该函数用于处理单个关键词在 pdf 注释部分的脱敏。

**Annot\_Ascii 函数:** 对英文关键词脱敏。

**Annot\_Unicode 函数:** 对中文关键词脱敏。

### 3.5 本章小结

本章在研究基于逻辑结构的 PDF 文件解析流程的基础上, 考虑到 PDF 文件逻辑结构的复杂性, 对象种类的多样性, 在敏感词脱敏的需求下实现难度大的情况下, 因此在无需考虑逻辑结构和敏感词脱敏的需求下, 提出了一种基于 Stream 流文本内容的提取方法。由于本文是对 PDF 文档的文本内容进行脱敏操作, 其注释和正文文本格式的差异性, 针对这两种情况设计和实现其文本内容提取方法; 考虑到提取出的 PDF 文件文本内容的编码方式尤为复杂, 提出采用循环匹配方式, 对可能存在的敏感词的组合方式方法进行匹配, 以达到良好脱敏的效果。

## 第 4 章 模式匹配算法

模式匹配是数据结构中字符串的一种基本的运算，在网络入侵检测、生物序列数据库比对（DNA）、信息检索、生物计量学等领域得到了广泛应用<sup>[45-48]</sup>。模式匹配按同时匹配模式串的个数分为单模式匹配和多模式匹配，其中 KMP、BM 为经典单模式匹配，经典多模式匹配有 AC<sup>[49]</sup>、WM<sup>[50]</sup>算法。PDF 文件的文本内容脱敏关键在于 PDF 文件被用户接收到后，其中的敏感信息已经被脱敏；因此在对 PDF 文件解析后，对敏感词的匹配准确性及高效性尤为重要，这时模式匹配算法就起关键作用，模式匹配算法的高效性及准确性是模式匹配算法的重点。本章首先介绍经典的单模式匹配算法，综合考虑 BM 算法和 QS 算法的特点，结合 PDF 文件格式和其编码规则，本章提出一种适用于 PDF 文件高效匹配算法，以支持涉密 PDF 文件内容的高效脱敏处理应用。

### 4.1 经典单模式匹配算法

迄今为止，国内外学者在 KMP、BM 等经典匹配算法的基础上，提出了不少改进的单模式匹配算法，如 BMH、QS、Q-Hash、TVSBS 等大量的精确单模式匹配算法。此外还存在大量利用移位表的计算规则、匹配方向的改进算法。这一节我们主要介绍经典的单模式匹配算法，如基于后缀比较的 BM 算法、基于下一字符的 QS 算法。

#### 4.1.1 相关概念介绍

模式匹配即给定字符集  $\xi$ ，给定文本串  $T$ ，长度为  $n$ ：

$$T = T_0 T_1 T_2 \cdots T_{n-1} \quad (4-1)$$

给定模式串  $P$ ，长度为  $m$ ：

$$P = P_0 P_1 P_2 \cdots P_{m-1} \quad (4-2)$$

在文本串  $T$  中匹配到给定的字符串  $P$ ，若字符串  $P$  出现在文本串  $T$  中：

$$T_{i+1} T_{i+2} \cdots T_{i+m-1} = P_0 P_1 \cdots P_{m-1} \quad (4-3)$$

则匹配成功且匹配的位置为  $i$ 。

#### 4.1.2 BM 算法

BM 算法是由 Robert S. Boyer 和 J. Strother Moore 于 1977 年共同提出的一种亚线性的单模式匹配算法。在实际应用中，BM 算法相对于其他算法，被普遍认为是一类高效算法。BM 算法的基本思想是基于后缀匹配，从右至左进行匹配，通过已获得的信息（移位表）跳过文本串中的某些字符，实现最大程度的移位。

BM 算法分为 2 个阶段，预处理过程和匹配过程。预处理阶段利用模式串信息构建移位表，即 BM 坏字符表（bmBc）和 BM 好后缀（bmGs），这是该算法的核心所在。

### 1. 预处理过程

预处理过程实质为坏字符表和好后缀表的建立。坏字符是指从左至右进行匹配时, 发生不匹配的字符被称为坏字符。好后缀是指模式串左边出现了模式串右侧的子串。以模式串“GCAGAGAG”为例, 构建坏字符表和好后缀表, 如表 4-1 和表 4-2 所示。

坏字符规则包含以下 2 种情况: 一是失配字符出现在模式串中, 这时需要把模式串是该字符移位对齐失配字符; 二是失配字符未出现在模式串中, 即无论怎么移位, 该字符都会产生失配, 即移位模式串至失配字符的下一位。

失配字符出现在模式串中, 移位如图 4-1。



图 4-1 有失配字符坏字符移位图

失配字符未出现在模式串中, 移位如图 4-2。

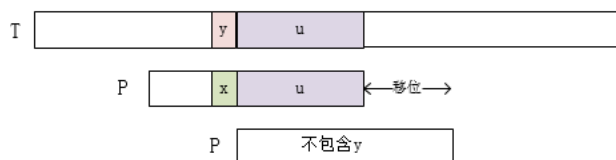


图 4-2 无失配字符坏字符移位图

坏字符移位计算如下:

$$bmBc[c] = \begin{cases} m & \text{字符未出现在模式串中} \\ m - j - 1 & j = \max\{j | P[j] = c, 1 < j < m\} \end{cases} \quad (4-4)$$

所以模式串坏字符计算结果如表 4-1:

表 4-1 BM 坏字符表				
坏字符 c	A	C	G	其他
bmBc[c]	1	6	2	8

好后缀的计算规则也包含以下 3 种情况: 一是模式串中左侧存在已匹配全部字符, 移位对齐已匹配的字符且前一字符不等于失配字符; 二是左侧存在已匹配的部分字符(后缀), 移位与之对齐; 三是左侧不存在已匹配字符, 移位到失配字符的下一位。

存在已匹配的字符且前一字符不等于失配字符, 如图 4-3 所示。



图 4-3 含好后缀且前字符不为失配字符移位图

存在部分已匹配的字符，如图 4-4 所示。

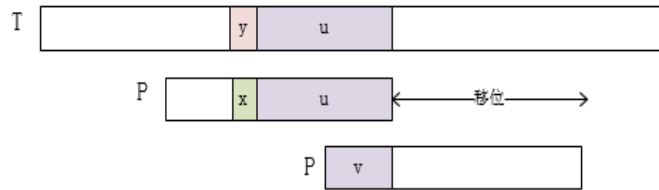


图 4-4 含部分好后缀移位图

不存在已匹配字符及其子串，如图 4-5 所示。

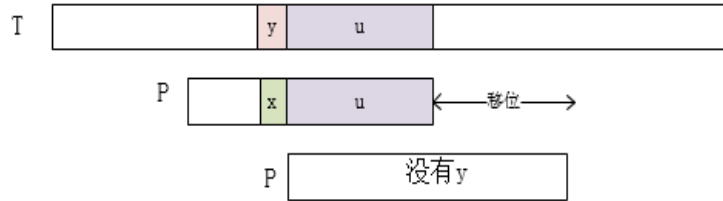


图 4-5 不存在好后缀及其子串

其好后缀的计算如下：

$$\text{bmGS} = \begin{cases} s & \text{与好后缀之间的距离 } s \text{ 是好后缀的最长子串} \\ m & \text{不存在好后缀及其子串} \end{cases} \quad (4-5)$$

所以模式例的好后缀表的计算结果表 4-2 所示。

表 4-2 BM 好后缀表

P[i]	G	C	A	G	A	G	A	G
bmGs[i]	7	7	7	2	7	4	7	1

## 2. 匹配过程

建立好后缀坏字符表后，从右至左开始匹配，一旦发生失配，利用预处理过程获得的好后缀和坏字符表，比较两个值的大小，取最大值进行移位，再进行下一次匹配，对已匹配成功的返回其在文本串中的位置。

## 3. 实例

给出的文本串=“GCATCGCAGAGAGTATACAGTACG”，模式串=“GCAGAGAG”，BM 算法在预处理过程分析中已得到其坏字符表  $\text{bmBc}$  和好后缀表  $\text{bmGs}$ ，其匹配过程如下。

第一次：

G C A T C G C A G A G A G T A T A C A G T A C G  
G C A G A G A G

$P[7] \neq T[7]$ ，查表得  $\text{bmGs}[7]=1$ ， $\text{bmBc}[A]=1$ ， $\text{bmGs} = \text{bmBc} + 8 - 8$ ，移位 1；

第二次：

G C A T C G C A G A G A G T A T A C A G T A C G  
G C A G A G A G

$P[5] \neq T[6]$ ，查表得  $\text{bmGs}[5]=4$ ， $\text{bmBc}[C]=6$ ， $\text{bmGs} = \text{bmBc} - 8 + 6$ ，移位 4。

第三次:

$$\begin{array}{c} \text{GCATCGCAGAGAGTATACAGTACG} \\ \text{GCAGAGAG} \end{array}$$

匹配成功, 查表得  $bmGs[0] = 7$ , 移位距离为 7。

第三次:

$$\begin{array}{c} \text{GCATCGCAGAGAGTATACAGTACG} \\ \text{GCAGAGAG} \end{array}$$

$P[5] \neq T[17]$ , 查表得  $bmGs[5] = 4$ ,  $bmBc[C] = 6$ ,  $bmGs = bmBc - 8 + 6$ , 移位 4。

第四次:

$$\begin{array}{c} \text{GCATCGCAGAGAGTATACAGTACG} \\ \text{GCAGAGAG} \end{array}$$

$P[6] \neq T[22]$ , 查表得  $bmGs[6] = 7$ ,  $bmBc[C] = 6$ ,  $bmGs > bmBc - 8 + 7$ , 移位 7, 匹配结束。BM 算法避免了 KMP 算法演变为 BF 的算法情况, 一般效率为 KMP 的 3-5 倍, 除在匹配方向上变化, 还引入了坏字符和好后缀表, 但是好后缀表的计算相对比较复杂, PDF 文本内容一个字是 4 个字符, 遇上了长敏感词, 好后缀处理就更加复杂。

#### 4.1.3 QS 算法

QS 算法是 Daniel M.Sunday 于 1990 年提出的一种快速单模式匹配, 也称 Sunday 算法。Sunday 通过对模式串扫描的顺序不同给出三种算法, 其中有 QS 算法, MS 算法、OM 算法<sup>[30]</sup>。MS 算法其扫描顺序为移位的大小降序, OM 算法的扫描顺序取决于其字符出现频率的大小。

QS 算法实质是 BM 算法的简化版本, 其实现更为简单, 仅利用了 BM 算法中的坏字符规则。其基本思想: 在进行字符串匹配时, 无论从左至右的方向, 还是从右至左方向进行匹配, 无需考虑匹配的顺序, 一旦发现不匹配, 至少移动一位, 即该匹配窗口的下一字符必考虑是否出现在模式串中, 利用 BM 算法的坏字符规则计算其移位, 再进行匹配。因此利用下一字符来实现最大跳跃, 使之最大位移为  $m+1$ 。

##### 1. 预处理过程

预处理过程实质是利用 BM 坏字符规则, 建立移位表。移位表建立也从字符是否在模式串及其位置考虑, 若下一字符不在模式串中, 则移位跳过该字符, 再在匹配窗口进行匹配; 若在, 移位与之对齐, 移位计算如下:

$$\text{移位}[c] = \begin{cases} m+1 & \text{字符未出现在模式串中} \\ m+1-j & j = \max\{j | p[j] = c, 1 \leq j \leq m\} \end{cases} \quad (4-6)$$

以上面模式串为例, 其移位表  $qsBc$  如表 4-3 所示。

表 4-3 QS 下一字符移位表

下一字符 a	A	C	G	其他
qsBc[a]	2	7	1	9

## 2. 匹配过程

建立好下一字符移位表后，无需像 KMP、BM 算法按顺序进行字符比较，即一旦发生不匹配，考虑该匹配窗口的后一字符，判断是否存在于模式串中，并移位继续进行匹配。

## 3. 实例

模式串和文本串和以上算法一样，预处理过程已经处理好 qsBc 数组，QS 无需考虑匹配方向，这里采用从右至左方式的匹配方向。

第一次：

```

GCATCGCAGAGAGTATACAGTACG
GCAGAGAG

```

$P[7] \neq T[7]$ ，这时直接考虑当前窗口的下一字符 G，查表  $qsBc[G]=1$ ，因此移位距离为 1；

第二次：

```

GCATCGCAGAGAGTATACAGTACG
GCAGAGAG

```

$P[4] \neq T[5]$ ，这时直接考虑当前窗口的下一字符 A，查表  $qsBc[A]=2$ ，因此移位距离为 2；

第三次：

```

GCATCGCAGAGAGTATACAGTACG
GCAGAGAG

```

$P[3] \neq T[6]$ ，直接考虑当前窗口的下一字符 A，查表  $qsBc[A]=2$ ，移位距离为 2；

第四次：

```

GCATCGCAGAGAGTATACAGTACG
GCAGAGAG

```

匹配成功，这时直接考虑当前窗口的下一字符 T，查表  $qsBc[T]=9$ ，因此移位距离为 9；

第五次：

```

GCATCGCAGAGAGTATACAGTACG
GCAGAGAG

```

$P[7] \neq T[20]$ ，这时直接考虑当前窗口的下一字符 C，查表  $qsBc[C]=7$ ，因此移位距离为 7，匹配结束。

一般情况下, QS 算法更适用于短模式和大字符集的情形下, 且实现简单。对于 PDF 文件文本内容来说, QS 算法相对于 KMP、BM 算法, 大多数情况都比较好; 但当出现下一字符移位距离小于失配字符的移位距离的情况, 将导致 QS 算法效率大打折扣。其次模式串太长, 且建立下一字符移位表可能就会考虑很多种情况, 并且没有考虑 PDF 文件编码格式, 最多移位距离  $m+1$ 。

## 4.2 改进算法

为了高效、准确地对 PDF 文档文本内容中的敏感信息脱敏, 脱敏成功关键在于敏感词的有效匹配, 也就是脱敏目标的确认。因此基于 BM 算法中的坏字符移位规则思想和 QS 算法的下一字符的思想, 结合 PDF 文件文本内容的编码特点, 我们提出了一种适用于 PDF 文档内容匹配的快速算法。

### 4.2.1 算法思想

在算法匹配时采用 BM 算法的后缀匹配, 即在模式匹配窗口中, 从右自左, 进行字符的比较。若模式串已经和文本串已成功匹配, 也利用 QS 的下一字符的思想, 利用该匹配窗口的下 4 个字符和模式串的前 4 个字符比较; 一旦发生不匹配, 必然会发生移位, 并且至少移动 4 位 (PDF 文本内容的一个字为 4 个字符), 因此利用 QS 算法的下一字符思想, 该匹配窗口的下 4 个字符可以考虑是否出现在模式串中, 再利用 BM 算法的坏字符计算规则, 得到移位表进行移位, 使最大移位为  $m+4$ , 提高了匹配的效率。

### 4.2.2 算法预处理及匹配过程

该算法和 BM 算法、QS 算法类似都包含 2 个过程: 先进行预处理过程, 处理字符移位情况的基础上, 进行模式串匹配过程。

#### 1. 预处理过程

预处理过程阶段, 主要实现移位表的计算, 即计算下四字符移位函数  $skip$ 。虽然移位函数计算是利用了 BM 函数的坏字符表的计算规则, 但是和 BM 坏字符表不同在于, 对于改进算法, 无需计算匹配串中的每一个字符的移位情况, 由于是从右至左开始匹配, 一旦发生不匹配, 模式串移位至少 4 位, 由于大字符集, 又是从右至左开始匹配, 所以考虑每一个字的最高位字符的移位表。

$$Skip = \begin{cases} m + 4 & \text{字符未出现模式串的最高位} \\ m - i - 1 + 4 & \text{其他} \end{cases} \quad (4-7)$$

#### 2. 匹配过程



通过预处理过程建立移位表信息后，就可以开始对模式串进行匹配。匹配过程如下：从右至左开始匹配，一旦不匹配，立刻利用该匹配窗口的下 4 个字符，判断其最高位的字符是否出现在模式串中的相应位置上，如果没有，说明这 4 个字符也不可能与模式串匹配成功，则直接移位模式串长度加上每个字的长度，即  $m+4$ ，否则移位模式串到其对应之处；若匹配成功，也考虑当前窗口的下四字符是否能对应模式串中以第一个字的最高位，若存在，移位  $m+4$ ，否则移位  $m$ 。

### 4.2.3 算法实现

具体算法实现代码如图 4-6 所示。

<p><b>预处理过程：</b>          输入：敏感词 <math>x</math>          敏感词的长度 <math>m</math>          输出：移位表 <math>bmBc</math>  <b>BEGIN</b>          FOR <math>i \leftarrow 0</math> to 256              DO <math>bmBc[i] \leftarrow m + 4</math>          END          FOR <math>i \leftarrow 0</math> to <math>m</math>              DO IF <math>(i + 1) \% 4 == 0</math>                  THEN <math>bmBc[x[i]] \leftarrow m - i - 1 + 4</math>          END  <b>END</b></p>	<p><b>匹配过程：</b>          输入：敏感词 <math>x</math>          文本串 <math>y</math>          敏感词的长度 <math>m</math>          文本串的长度 <math>n</math>          输出：敏感词的个数 <math>cnt</math>          比对次数 <math>cnt\_bd</math>  <b>BEGIN</b>          Calculate <math>bmBc</math>          WHILE <math>j &lt; (n - m)</math>              DO <math>cnt\_bd \leftarrow cnt\_bd + 1</math>              FOR <math>i \leftarrow m - 1</math> to 0 BY <math>x[i] == y[i + j]</math>                  DO <math>cnt \leftarrow cnt + 1</math>              END              IF <math>i &lt; 0</math>                  THEN IF <math>x[3] == y[m - 1 + 4 + j]</math>                      THEN <math>j \leftarrow j + m</math>                      ELSE <math>j \leftarrow j + m + 4</math>                  ELSE <math>j \leftarrow j + bmBc[y[m - 1 + j + 4]]</math>              END  <b>END</b></p>
--	--

图 4-6 改进算法代码实现

### 4.2.4 算法实例

以 PDF 文档为例，为了更清楚的识别每个字，每个字之前用空格隔开。我们考虑如下的文本串=“4e00 79cd 5feb 901f 7684 5355 6a21 5f0f 5339 914d 7b97 6cd5 5339 914d”。

模式串=“5339 914d”。

首先进行预处理过程，skip 表的建立。模式串每个字的最高为分别为 9、d，如表 4-4。

表 4-4 Skip 表

下 4 字符 a	9	d	其他
Skip[a]	8	4	12

第一次:

4e00 79cd 5feb 901f 7684 5355 6a21 5f0f 5339 914d 7b97 6cd5 5339 914d  
5339 914d

$T[7] \neq P[7]$ , 则考虑匹配窗口的下 4 个字符的高位 b, 字符 b 未在模式串相应的位置上出现, 此时  $T[11] \neq P[3] \neq P[7]$ , 这时移位 12 位。

第二次:

4e00 79cd 5feb 901f 7684 5355 6a21 5f0f 5339 914d 7b97 6cd5 5339 914d  
5339 914d

$T[19] \neq P[7]$ , 则考虑匹配窗口的下 4 个字符的高位 5, 字符 5 未在模式串相应的位置上出现, 此时  $T[23] \neq P[3] \neq P[7]$ , 这时移位 12 位。

第三次:

4e00 79cd 5feb 901f 7684 5355 6a21 5f0f 5339 914d 7b97 6cd5 5339 914d  
5339 914d

$T[32] \neq P[7]$ , 则考虑匹配窗口的下 4 个字符的高位 9, 发现字符 9 在模式串相应的位置上出现,  $T[35] = P[3]$ , 这时移位 8 位。

第四次:

4e00 79cd 5feb 901f 7684 5355 6a21 5f0f 5339 914d 7b97 6cd5 5339 914d  
5339 914d

发现匹配, 考虑匹配窗口的下 4 个字符, 发现字符 7 未出现在模式串相应的位置, 即移位后不可能出现匹配, 则移位 12 位。

第五次:

4e00 79cd 5feb 901f 7684 5355 6a21 5f0f 5339 914d 7b97 6cd5 5339 914d  
5339 914d

$T[51] \neq P[7]$ , 则考虑匹配窗口的下 4 个字符的高位 d, 字符 d 在模式串相应的位置上出现,  $T[55] = P[7]$ , 这时移位 8 位, 匹配成功并结束。

改进算法与 QS 算法一样, 可能存在一个相同的问题, 即当下 4 个字符出现在模式串中, 且可能在靠模式串右边一点, 但该匹配窗口的最右端的字符并未出现在模式串中, 这个时候只根据下 4 字符判断移位, 可能会使移位距离减小, 增加了比较的次数。因此可考虑利用 BMH 算法的最后一个字符的移位表, 和下 4 字符的移位表进行比较, 选择最大移位。在某些情况下, 选两者进行比较, 可能会提高效率。

### 4.2.5 测试结果分析

模式匹配需要高效的在文本串中找到模式串，为更好对以上改进算法进行性能分析，因此对其整个匹配过程所需时间进行测试。

实验环境为 win7 64 位操作系统，配置为 Intel (R) Core (TM) i7-2670QM CPU @2.20Ghz 2.19Ghz，内存为 4GB，C 语言编程。由于在对 PDF 文件进行解析时，过程相对复杂，这里实验文本采用模拟 PDF 文本内容的文本进行实验，其文件大小为 8.76M，内容为 2296856 个汉字，即 9187424 个字符，对其分别进行字符长度为 4、8、12、16、20、24、28、32、36、40 的模式串匹配。为减小误差，针对不同模式串的长度，分别对每种匹配算法测试 10 次，取其平均值，其测试结果如表 4-5。

表 4-5 不同模式串长度、不同算法 CPU 耗时 (ms)

长度 算法	4	8	12	16	20	24	28	32	36	40
BM	39.1	22.65	18.3	15.75	12.33	12.4	11.1	11.6	10.7	10.2
QS	22.8	17	14	12.9	8.9	9.2	9.7	11.3	9.2	7.42
改进	12.5	13.2	9.55	8.95	6.77	6.4	6.7	6.2	4.98	4.55

由表 4-5 我们可以看出，通过改进的适合 PDF 文本内容的匹配算法，相对于 BM、QS 算法，改进算法更为高效，其时间复杂度更小。其搜索时间对比如图 4-7 所示。

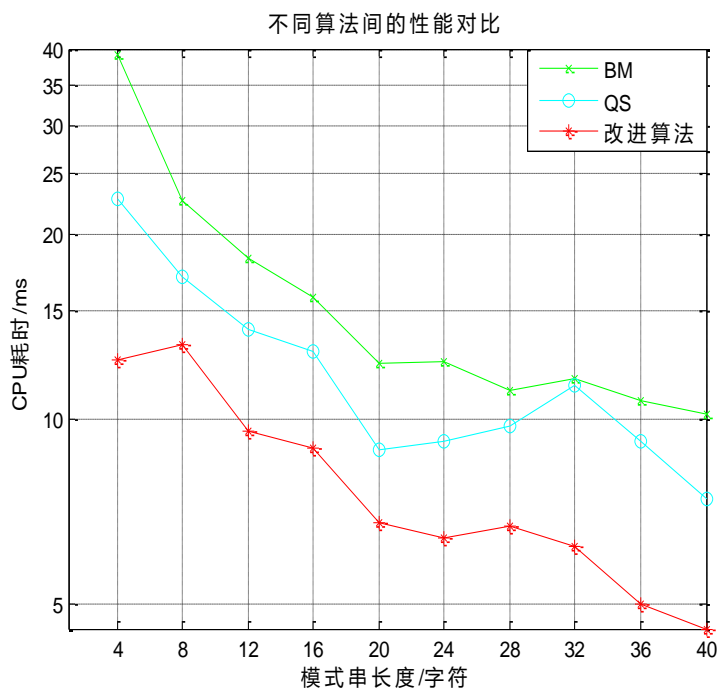


图 4-7 PDF 文档内容搜索时间对比图

针对不同长度的模式串，对 BM、QS、改进算法的匹配次数进行比较，如图 4-7 所示。

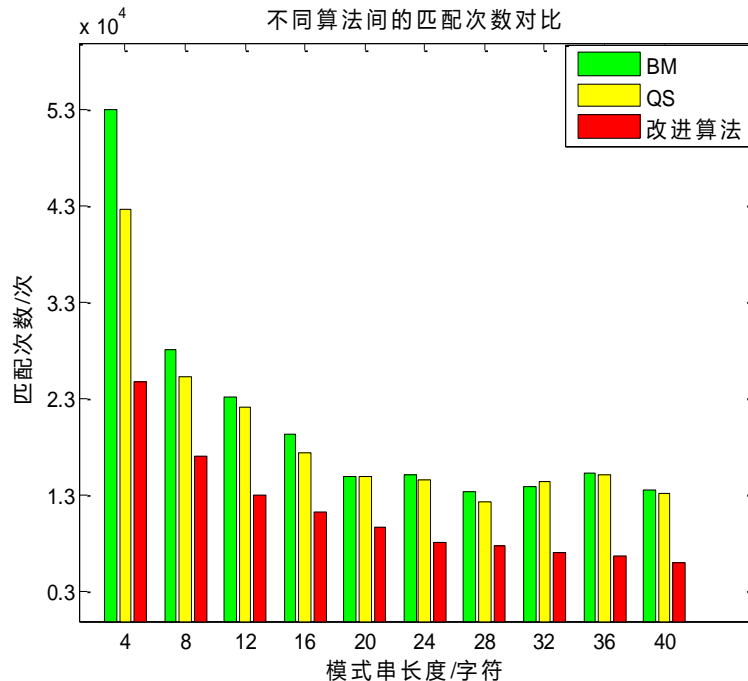


图 4-8 PDF 文档内容匹配次数对比图

匹配算法效率跟很多因素有关，如匹配串长度、大字符集等。其中，QS、BM 算法匹配效率与匹配串长度密切相关，由于其最大移位距离与模式串的长度有关，其最大移位距离分别为  $m+1$ 、 $m$ ；其中 QS 是 BM 的简化版，实现简单，不过存在下一字符在模式串中导致移位距离相对于 BM 减小的情况。由图 4-8 我们可以看出，相比于经典算法，改进后的算法时间耗时和匹配次数最小，基本上随长度的增加而减小。

除与经典算法 BM 和 QS 算法做比较外，文章针对近年在基于 BM 和 QS 算法的部分改进算法也做了相关比较。如：文献[51]基于 BM 算法提出 SBM 算法，其从匹配方向入手，从两边往中间匹配，出现不匹配则利用下一字符的移位的算法；文献[52]基于 BM、BMH 提的改进算法（这里为区别文章的改进算法，以 YBM 算法替代），该算法从移位表入手，发生不匹配时，考虑末字符和下一字符的移位距离，选择最大进行移位；文献[53]基于 BM 等算法，提出 BMQ 算法，其利用末字符及下一字符组合性，其实质为同时考虑移位后的末字符来增加移位；还有基于 QS 的 EQS 和 IQS 算法<sup>[54-55]</sup>，其很相似，不同在于 EQS 利用当前窗口的后 2 字符确定移位，而 IQS 由后 3 字符确定移位。

在测试环境、条件相同的条件下，论文比较了不同模式串长度下不同算法的 CPU 耗时，其测试结果如表 4-6 所示。

表 4-6 不同模式串长度、不同算法 CPU 耗时 (ms)

长度 算法	4	8	12	16	20	24	28	32	36	40
SBM	25.4	17.7	16.13	11.5	11.8	9.17	8.22	11	11.02	8.15
YBM	32.14	20.56	17.34	14.11	12.32	9.5	9.7	11.51	11.2	10
BMQ	25.1	22.21	18.1	13.2	12	11.3	11	12.33	10.6	9
EQS	22.8	17	14	12.9	8.9	9.5	9.7	11.3	9.8	10.1
IQS	25	23.7	22.3	15.5	13.55	9.2	9.02	10.4	9.2	7.85
改进	12.5	13.2	9.55	8.95	6.77	6.4	6.7	6.2	4.98	4.55

由表 4-6 可见, 本文所提出的改进算法在特定情况下, 比经典算法和其多种衍生算法在 CPU 耗时上有优势, 这表明本文所提出的改进算法计算效率更高。为更直观的分析 and 对比, 分别对改进 BM、QS 算法做比较, 如图 4-9、4-10 所示。

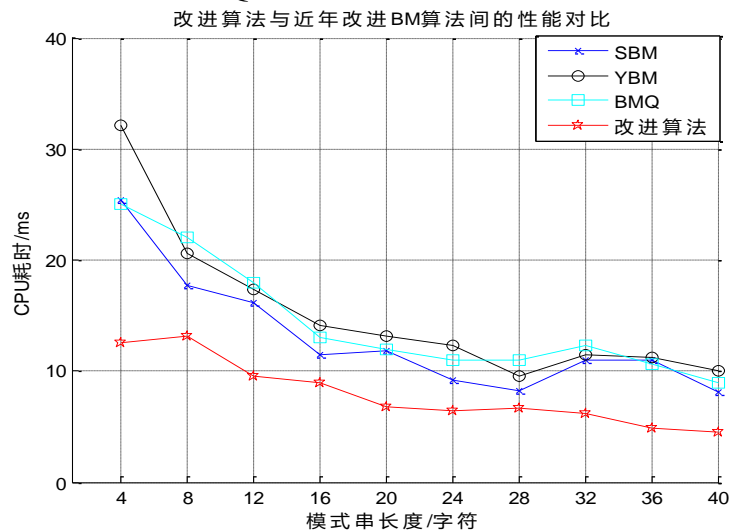


图 4-9 改进 BM 算法间的搜索时间对比图

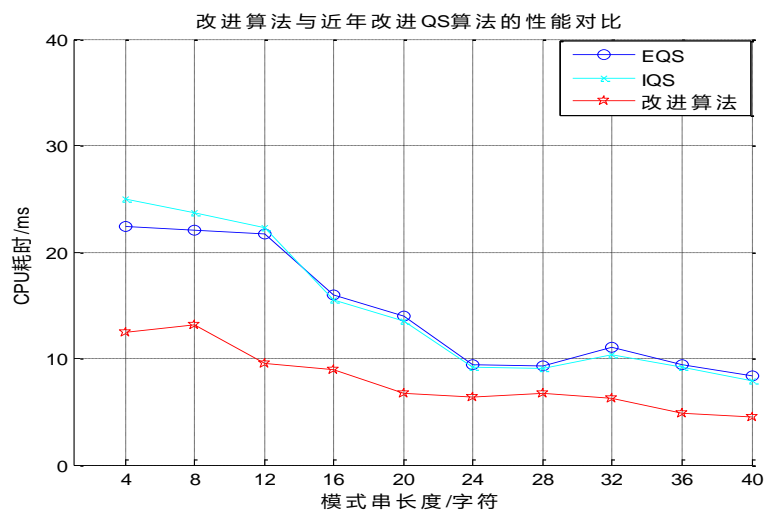


图 4-10 改进 QS 算法间的搜索时间对比图

从图 4-9 和 4-10 可以看出, 基于 PDF 文本编码特征的改进算法其搜索时间明显小于近年改进算法, 不过算法的效率几乎都与模式串长度有一定的关系。

此外还针对不同长度模式串, 对以上算法做了字符串匹配次数比较, 如图 4-11、4-12 所示。

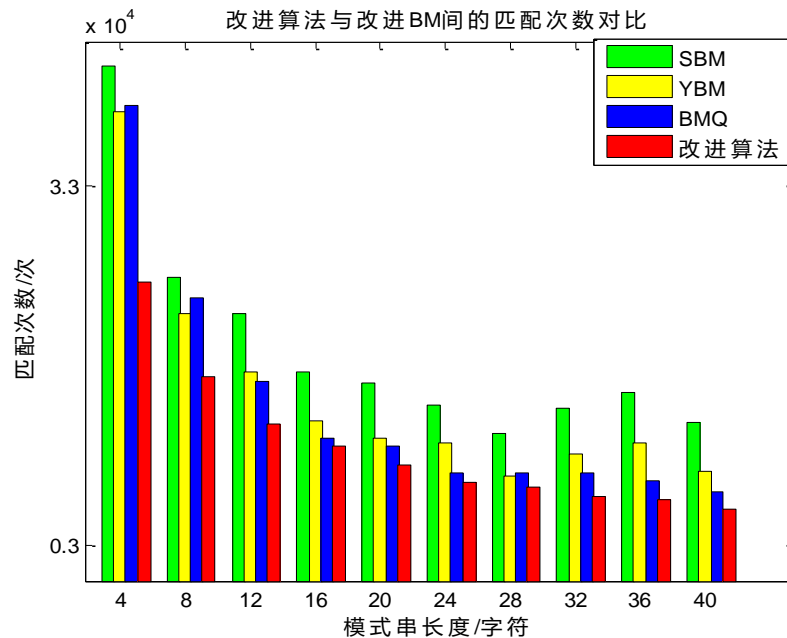


图 4-11 改进 BM 算法间的匹配次数对比图

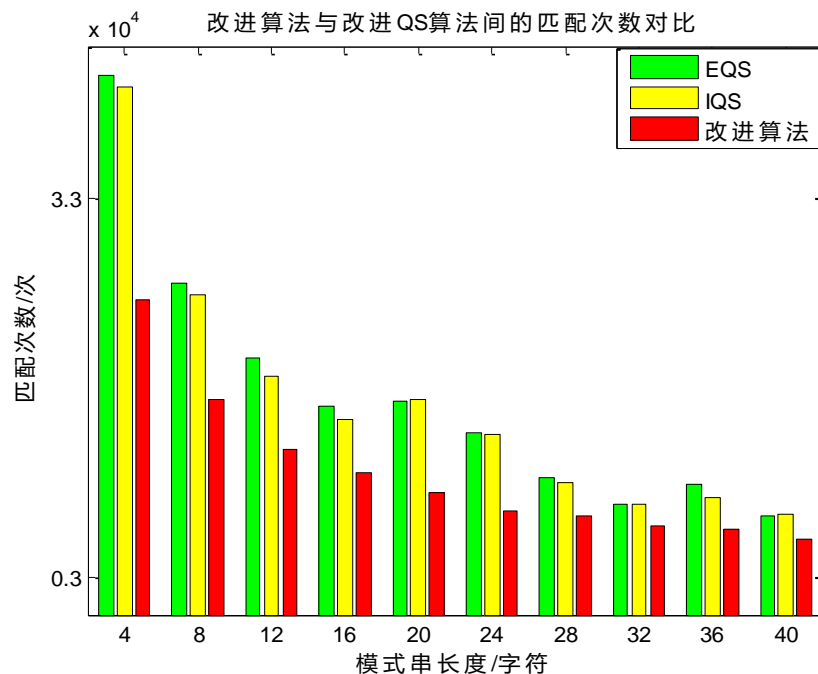


图 4-12 改进 QS 算法间的匹配次数对比图

从图 4-11 和 4-12, 可以看出, 本文的改进算法匹配次数明显小于其他算法, 从图 4-9 至 4-12, 可以看出, 虽然某些算法的耗时相对高于另一个, 但是其匹配次数却相对

小，这是因为某些算法追求的是通过增加跳跃距离、减少匹配次数来提高效率，但某些算法实现又为复杂，有利有弊。但本文提出的改进算法基于 PDF 文档编码格式，其效率和匹配次数等表现明显。

最后，针对改进算法，对模式串占比情况进行匹配次数和时间耗时的测量，如图 4-13、图 4-14 所示。

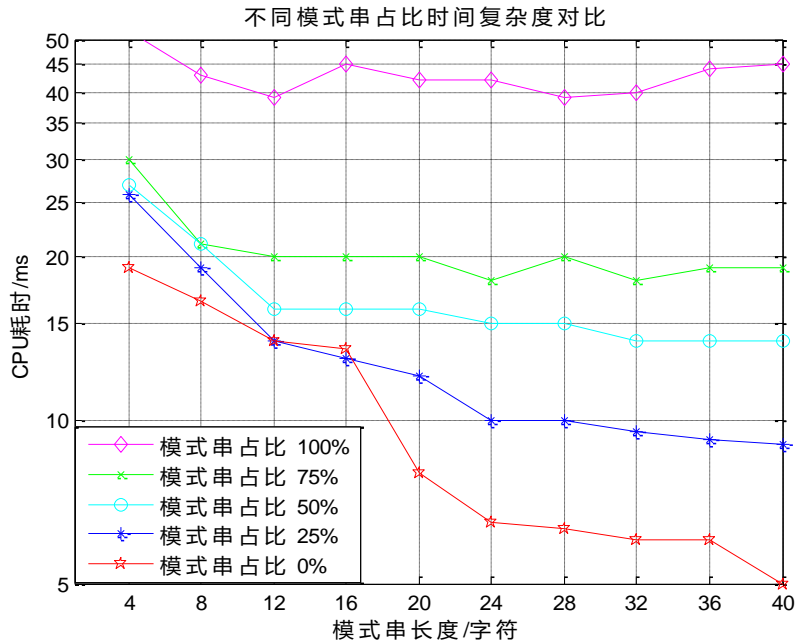


图 4-13 PDF 文档内容搜索时间对比图

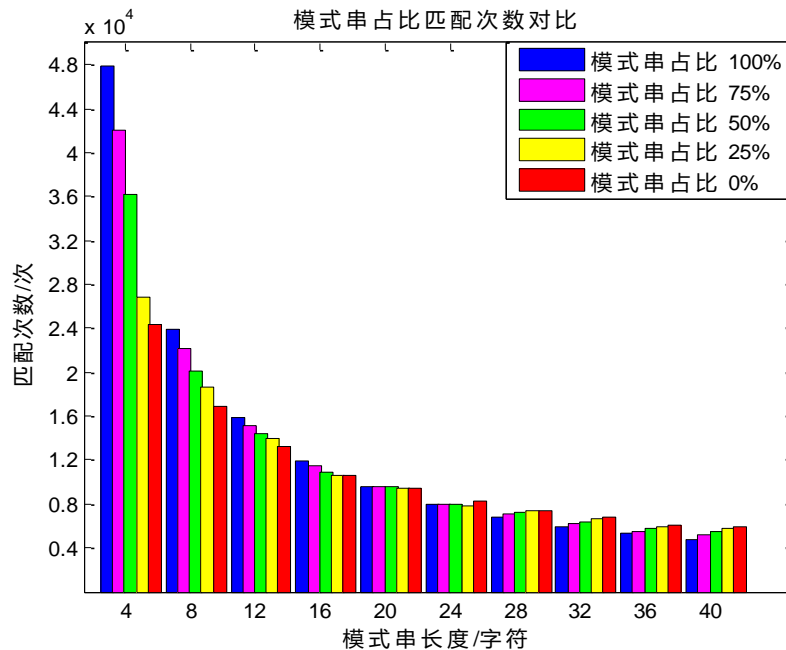


图 4-14 PDF 文档内容匹配次数对比图

由图 4-13 和 4-14 我们可以看出，模式串长度越大、模式串占比越小，时间耗时越小；此外，从图中看出，当模式串长度达到一定长度时，匹配次数随占比的减小而增大，可能因为下四字符出现在模式串的概率增大，而减小了移位，导致了比对次数的增加，但是总体性能都比较好。

改进的算法在 BM、QS 的基础上，结合 PDF 文本内容编码方式，只生成每个字高位的移位距离，使最大移位距离为  $m+4$ ，由图可知，其匹配效率比 KMP、BM、QS 以及近年部分改进算法都高，该算法应用于 PDF 文本环境中后，其性能有一定的提高。

### 4.3 本章小结

在本章中，针对 PDF 文件解析后，对文本内容进行脱敏处理，为提高效率，以敏感目标确认为方向，深入研究模式匹配算法，在 Robert S. Boyer 和 J.Strother Moore 提出的 BM 算法和 Sunday 提出的 QS 算法基础上，结合 2 种算法的优点，利用 PDF 文件文本内容的编码规则，提出适用于 PDF 文件内容的高效字符串匹配算法。通过对比前人提出经典的匹配算法 BM、QS 算法以及部分改进算法后，在保证准确性一样的情况下，本文提出改进算法的有良好的效率。针对网络中传输的 PDF 文件，要求其识别敏感目标的准确性及高效性，提出的改进算法都能很好的满足。



## 第 5 章 脱敏系统的框架设计与实现

针对网络中传输的 PDF 文件,本章主要讨论对在线 PDF 文件文本内容进行实时的有效监控。本文给出脱敏系统的总体框架设计与具体实现方案,以实现 PDF 文件截获、内容解析分析和脱敏处理。

### 5.1 HTTP 协议

#### 5.1.1 HTTP 协议简介

HTTP (Hyper Text Transfer Protocol), 即超文本传输协议, 是面向应用层的协议, 因其具有简单快速、灵活、支持客户/服务器模式(由请求和响应构成)、无连接、无状态等特点, 成为应用最广泛的一种协议。经多年发展, 已经演化了多个版本, 从 0.9 版到 1.1 版, 目前由万维网协会 (World Wide Web Consortium) 和互联网工程工作小组 (Internet Engineering Task Force) 合作制定的 HTTP 1.1 版被默认使用, 但其 HTTP 1.0 版目前在代理服务器中仍被广泛使用<sup>[56]</sup>。由于 HTTP 协议的广泛应用, 基于 HTTP 协议的 Web 应用、服务端、客户端等也容易成为了攻击目标<sup>[57]</sup>。

一次 HTTP 操作作为一个事务, 事物的发展均有开始和结束。一个事务也要先建立 TCP 连接, 经过一系列的 HTTP 操作后再释放 TCP 连接, 其 HTTP 协议会话过程流程如下<sup>[58]</sup>:

- 1) 建立 TCP 连接, 即实现三次握手;
- 2) 发送请求。建立连接后, 客户端向服务器发送请求, 常用的 HTTP 报文请求方法有 GET (请求获取由 request-url 标识的资源)、POST (在 request-url 所标识的资源后附加新的数据)、HEAD、TRACE、PUT、DELETE、OPTIONS, 常用的一般是 GET 和 POST;
- 3) 发送响应。服务器收到来自客户端的请求信息后, 便进行处理, 回发相应的响应信息给客户端;
- 4) 释放 TCP 连接, 即实现四次挥手过程。

HTTP 的一次请求响应模型如图 5-1 所示。

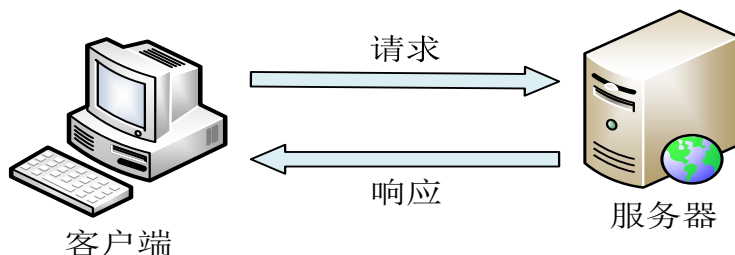


图 5-1 HTTP 请求响应模型

当客户端通打开应用程序 web浏览器，在地址栏输入一个URL，这是浏览器将根据请求创建并发送一个请求，该请求中包含URL和浏览器自身的相关信息。当服务器收到这个请求时，将返回一个响应，其中包括与该请求相关的信息以及指定URL（如果有）的数据。

5.1.2 HTTP 报文请求格式

HTTP请求报文由3部分组成，其中包含请求头、请求行、请求正文，其格式如图 5-2所示。

请求方法	URL	协议版本	<request-line>
头部字段：值	<headers>		
回车换行	<blank line>		
请求正文	<request-body>		

图 5-2 请求报文格式

常用的请求方法在上一小节已提及。请求头部由键值对组成，其键值用“冒号”隔开，向服务器传递客户端的一些附加信息。常用的请求报头的相关介绍如表5-1。

表 5-1 常用请求报头

头部字段	类型	描述
Accept	通用	客户端可以接受的内容的类型，如 text/html、application/pdf
Accept-Encoding	实体	客户端能处理的压缩类型，如 gzip、deflate
User-Agent	请求	请求浏览器的类型
Cookie	实体	HTTP 请求发送时，会把保存在该请求域名下的所有 cookie 值一起发送给服务器，即标识信息，最重要头信息之一
Connection	通用	Connection: close 响应结束之后，连接会被关闭。若应用程序不支持持久连接，则需加入 Connection: close
Host	请求	初始 URL 中的主机和端口，即请求资源所在的服务器
Range	请求	指定客户端可以需要 Range 后指定的数据位置
If-Range	请求	如果实体未改变，服务器只发送丢失的部分，否则发送整个实体。

5.1.3 HTTP 报文响应消息格式

HTTP响应报文也由3部分组成，其中包含响应头、响应行、响应正文，其格式如

图5-3所示。

协议版本	状态码	状态码描述	<status-line>
头部字段: 值			<headers>
回车换行			<blank line>
响应正文			<response-body>

图 5-3 响应报文格式

状态行（status line）提供所请求的资源情况，从上图5-3看出，由协议版本、状态码和状态码描述构成。协议版本如HTTP/1.1；状态码由3位数构成，分为以下5类：第一位数代表响应的类型，如1xx：表示请求信息已收到；2xx：表示成功并接收处理；3xx：用于重定向；4xx：指示客户端的错误；5xx：指示服务器端的错误。此外，响应报文的头部信息和请求报文的头部信息相似，也是为响应报文添加一些附加信息，常用的响应报头的相关介绍如表5-2所示。

表 5-2 常用响应报头

头部字段	类型	描述
Date	通用	标识报文创建的时间和日期
Content-Length	实体	实体内容的长度，客户端通过它确定报文截尾
Content-Type	实体	报文中对象的媒体类型，如image、text、pdf
Cache-Control	通用	见 “缓存首部”
Content-Encoding	响应	服务器可支持的压缩编码类型
Content-Range	响应	整个返回体中本部分的字节位置
Transfer-Encoding:chunk	响应	使用分块传输
Content-Language	响应	响应体使用的语言
Location	响应	用来重定向接收方到非请求URL，以此来完成请求或标识新的资源
Vary	响应	其值是个首部列表，通过协商，服务端去查询哪些是客户端会使用的首部，并去查看，以决定以什么内容返回给客户端

## 5.2 总体架构

实际应用中，与访问权限等级、敏感信息防护或实时监控相关时，一般可以采用动态脱敏。目前，动态数据脱敏有基于代理实现机制、基于视图实现两种机制<sup>[16]</sup>。两者相比，基于代理的脱敏灵活性、适应性较好。一般有常见三种代理服务器（proxy server）：正向代理服务器、透明代理服务器、反向代理服务器<sup>[59]</sup>。

但是本文目的在于对网络中的 PDF 文件进行脱敏操作，让客户端接收到的文件是经脱敏处理后的文件，反向代理代理的对象是服务器，因此采取基于反向代理机制的脱敏系统。脱敏过程对于客户端是完全透明的，也就是客户端感觉不到存在着脱敏处理，并且脱敏的实现是在数据容器外，这更适用于非结构化数据，如 PDF 文件。

在实际应用中，一般是对网络中的 PDF 文件进行脱敏处理，防止敏感信息的快速扩散，或者用于舆情的监测等。本文 PDF 文件脱敏代码安装于脱敏服务器（代理服务器），在网络中，整个脱敏系统拓扑示意如图 5-4 所示。



图 5-4 网络拓扑图

图中用户相当于 C/S 架构的客户端（Client），其应用程序一般为 Web 浏览器，通过其地址栏输入内容服务器的域名或 IP 地址对其资源进行访问；图中脱敏服务器为 PDF 文件解析及内容脱敏的载体，在网络与拓扑结构中，脱敏服务器放在用户和内容服务器之间，该位置类似于网关的位置；此时，可根据内容服务器的 IP 地址判定是否对该数据包进行处理（包过滤），再进行敏感信息脱敏操作；图中内容服务器为 C/S 架构的服务端（Server），其存放具体的内容，如 PDF 文件，等待用户的访问。

此外，可以从网络拓扑图来理解基于反向代理的脱敏系统，该系统的实质就是脱敏服务器充当反向代理服务器。即当用户通过浏览器请求一个 PDF 文件，由脱敏服务器代访问，从内容服务器处获得对应的 PDF 文件，这时进行 PDF 文件解析、敏感词确认及启动脱敏机制，再返回给用户，即用户收到的 PDF 文件已经是经过脱敏处理后的文件，其原理图示意如图 5-5 所示。

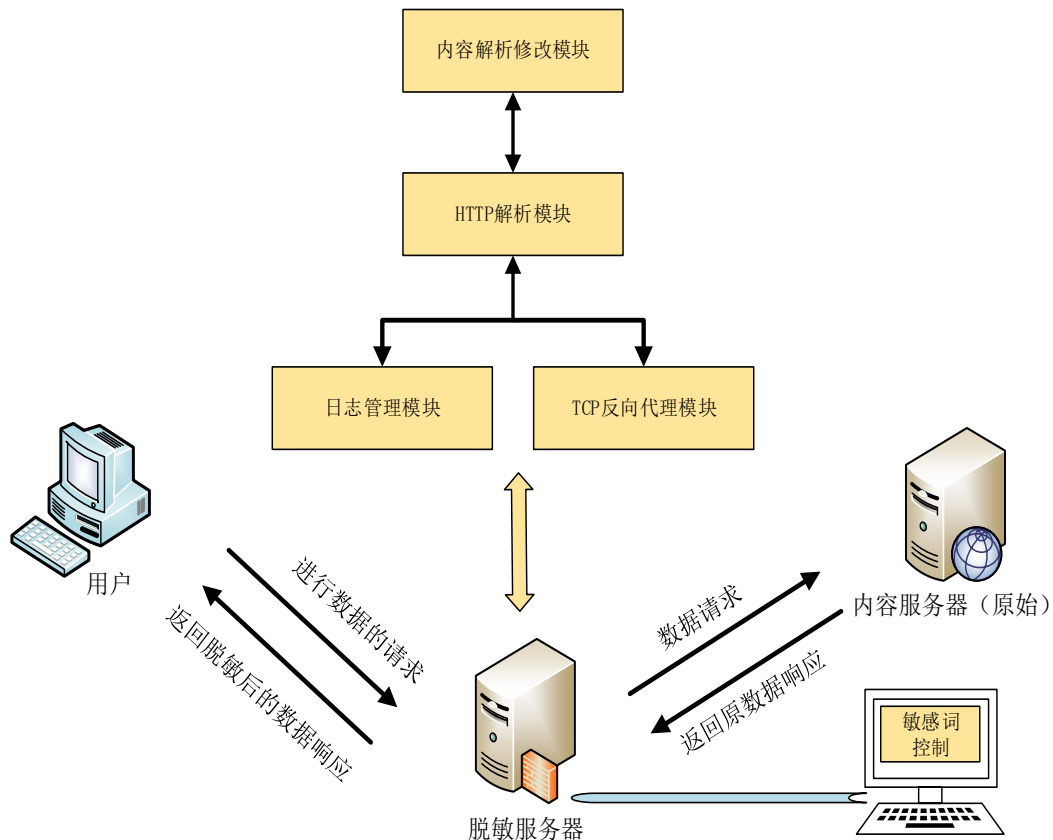


图 5-5 脱敏系统原理示意图

如图所示，脱敏系统主要是由 TCP 反向代理模块与日志模块、HTTP 协议解析模块和内容解析模块实现，其工作流程如下：

- 1) 脱敏服务器启动 TCP 反向代理，通过本地 5500 端口代理内容服务器的 80 端口；
- 2) 用户通过浏览器向脱敏服务器 5500 端口发送 PDF 文件请求；
- 3) 请求的数据被脱敏服务器收到后，交给 TCP 反向代理模块、HTTP 解析模块进行处理（如删除某些 HTTP 请求报文的头部字段信息），再由 TCP 反向代理模块发往内容服务器；
- 4) 内容服务器收到请求报文后，进行响应，将请求到的数据（如 PDF 文件）作为响应数据包返回给脱敏服务器；
- 5) 响应数据被脱敏服务器收到后，通过 TCP 代理模块进行 TCP 层解析，提取其数据部分；再递交给 HTTP 解析模块，对其进行 HTTP 层解析，获得真实数据部分（含 PDF 文件的内容），在监测到是 PDF 文件通过脱敏服务器时，递交给内容解析模块，进行 PDF 文件解析和脱敏处理；
- 6) 内容解析模块收到须处理的数据信息时，会触发 PDF 文件解析及脱敏工作，完成敏感信息脱敏工作，并将脱敏完成的后的脱敏结果递交给日志模块对其进行记录；

- 7) 最后依次返回操作，由代理模块将脱敏完成后的数据包重新封装好后发往用户，用户收到后还原出是脱敏处理后的 PDF 文件。

### 5.3 基于反向代理的脱敏系统模块的设计与实现

#### 5.3.1 TCP 反向代理模块

连接管理由 TCP 反向代理模块和日志管理模块组成，其中反向代理模块模块是基于 boost::asio 网络库设计，而日志管理模块则依赖于 easylogging++ 库，TCP 反向代理模块主要完成以下操作，如图 5-6 所示。

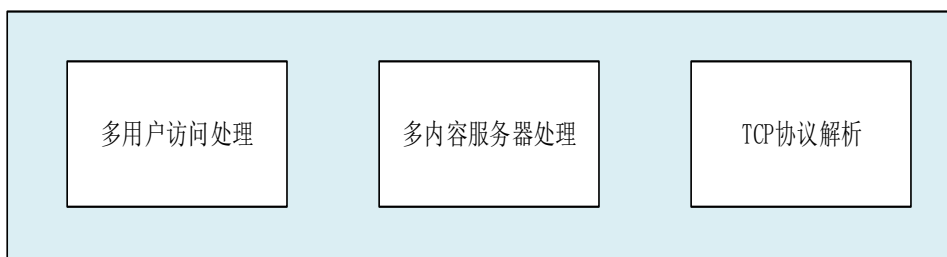


图 5-6 TCP 反向代理模块

TCP 反向代理模块通过异步通信的方式来管理多个客户端的同时访问与响应，其底层设计是基于 Linux 下 epoll 机制的封装，从而达到 Proactor 模式的效果。对响应 TCP 数据进行存储，传给上层程序处理，通过上层返回值确定是否继续存储，并在上层程序处理结束后打印异常到日志系统中。TCP 反向代理模块建立在 linux 套接字 (socket) 的基础上，通过套接字来发送接收数据，其流程为：首先，实例化一个 acceptor，建立一个会话连接；接着调用 accept 异步等待客户端连接，连接一旦到达即开启客户端和代理之间的异步 socket 通信以及远端服务器和代理之间的异步 socket 通信；再次建立会话连接，accept 异步等待新连接，从而形成循环，保证服务器一直处于运行状态，流程图如图 5-7 所示。

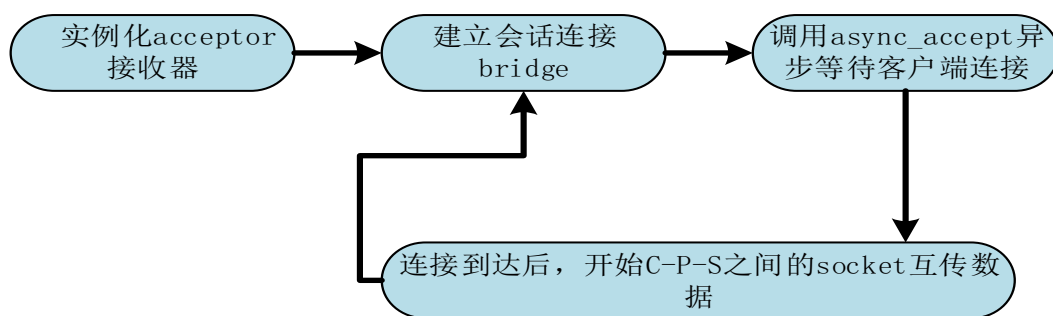


图 5-7 反向代理设计流程图

当脱敏服务器收到内容服务器发来的响应数据报文时，需检测收到的数据包是否完整，完整时才返回给上层处理；其次考虑释放掉前面传来的数据包缓存，重新接收，其处理流程图如 5-8 所示。



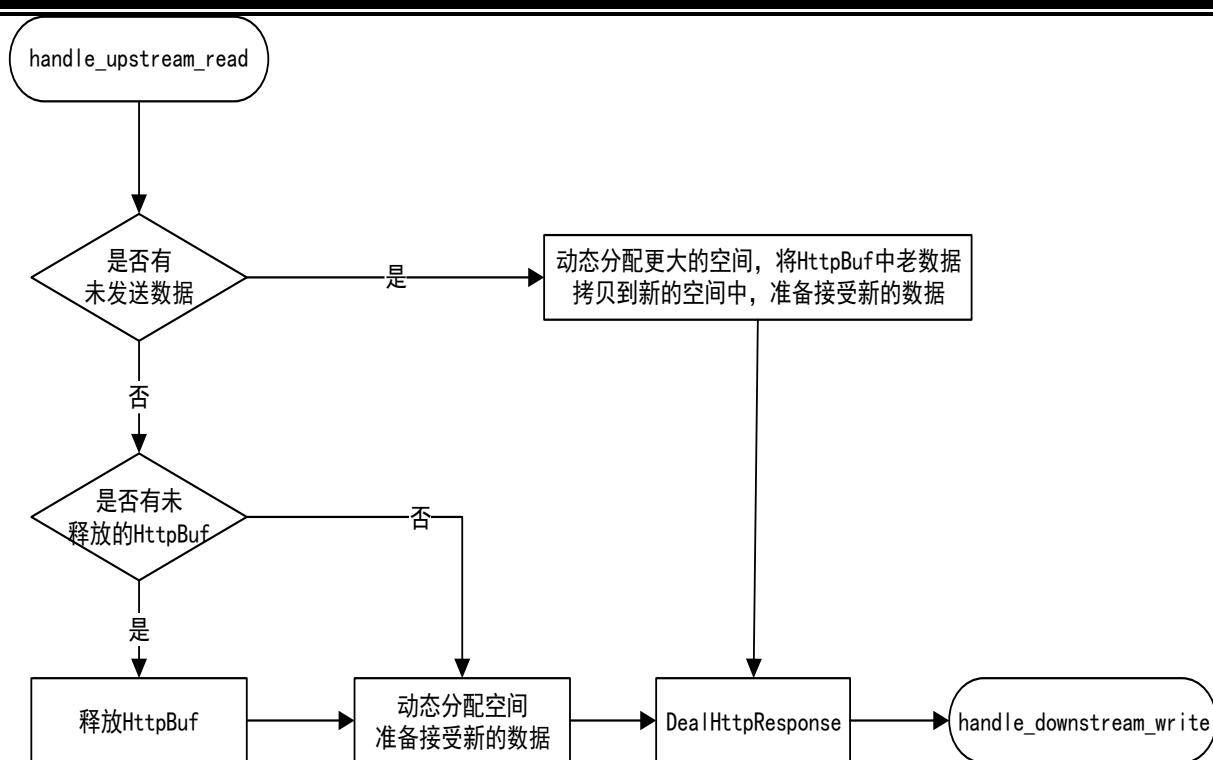


图 5-8 响应数据接收处理流程图

### 5.3.2 HTTP 协议解析模块

HTTP 解析模块在应用层实现，对截获的 TCP 报文进行处理，对 TCP 报文的数据部分进行提取并分析，其数据部分是 HTTP 协议传输的报文。该模块分别对 HTTP 的请求报文和响应报文进行解析，建立在 TCP 反向代理和日志模块的基础之上，通过 TCP 反向代理模块发送接收数据。HTTP 解析模块主要完成以下操作如图 5-9 所示。

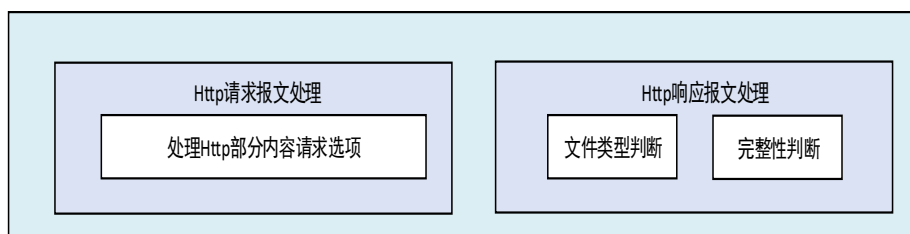


图 5-9 HTTP 协议解析模块

如果 HTTP 报文是请求报文，对其头部字段信息进行判断，确认是否包含“Range:... \r\n”和“If-Range: ....\r\n\r\n”字段（因客户端存在缓存数据，只接受部分 HTTP 数据文件），若存在该字段，为避免某些 pdf 文件不完整，而无法成功脱敏，所以将该字段从头部信息中删除。

如果 HTTP 报文是响应报文，判断该报文头部字段信息是否有“Content-Type:”字段，如果有，代表当前数据包的应用类型；若该字段后的值为“application/pdf”，代表传回的数据是一个 pdf 文件。通过对 HTTP 报文的解析，可以获得当前 HTTP 报文

的数据类型，进而对该数据信息（PDF 文件）中的敏感信息进行处理，完成脱敏操作。其中，利用快速 QS 匹配算法，判断是否为 PDF 文件，主要函数代码实现如下：

```
QS(PtoContentType,const_cast<char*>("application/pdf"),end-PtoContentType,1);
```

此外针对某些静态的 PDF 文件，即如果当前的 HTTP 报文是静态文件类型，那么 HTTP 响应报文的头部字段信息中包含“Content-Length”字段，代表了当前数据包的长度。因为某些静态 PDF 文件，必须在文件完整的情况下，才能完成脱敏操，因此需对该字段的值与 Buf 中的数据长度进行比较，以此来判断当前 Buf 中的 PDF 文件数据是否完整；若该数据信息完整，则进行 PDF 文件的脱敏处理；若不完整，则重新获取 Buf 的数据，直到接收完整，代码具体实现如图 5-10 所示。

```
if( ((PtoPdfLength = QS(*buf, const_cast<char*>("Content-Length: "), end-start, 16)
) != NULL) ){
    ..... //找到其开始结束的位置
    char *PdfHttpTemp = KMP(PtoPdfData, const_cast<char*>("HTTP/1."), *bufsize-
(PtoPdfData-*buf), 7);
    //该buf中含有多个HTTP报文数据，其中第一个pdf文件
    if(PdfHttpTemp != NULL) {
        //判断该pdf文件处于完整状态
        if(Pdflength == PdfHttpTemp-PtoPdfData) {
            //对第一个HTTP报文中的pdf文件进行文件脱敏处理
            int err = pdf(PtoPdfData, Pdflength, KeyWord_word);
            ..... //日志管理
            //将之后的http报文发回缓冲区
            *bufRes = PdfHttpTemp;
            return 2;
        }
    }
    //该buf中只有一个HTTP报文
    else {
        //判断pdf文件处于完整状态
        if(Pdflength == *bufsize - (PtoPdfData-*buf)){
            //进行pdf文件脱敏处理
            int err = pdf(PtoPdfData, Pdflength, KeyWord_word);
            ..... //日志管理
            *bufRes = NULL;
            return 1;
        }
        //该buf中的pdf文件不完整，返回0，将buf发回缓冲区
        else{
            *bufRes = *buf;
            return 0;
        }
    }
}
```

图 5-10 判断 PDF 文件完整代码

HTTP 协议解析的主要流程主要由 parseHttp（）函数和 DealRequest（）函数实现，parseHttp 函数的主要功能是对内容服务器传回的 HTTP 包判定，其是否为 PDF 文件并进行脱敏；DealRequest 函数主要是处理 HTTP 请求报文，删除指定的头部信息。其流解析程如图 5-11 所示。



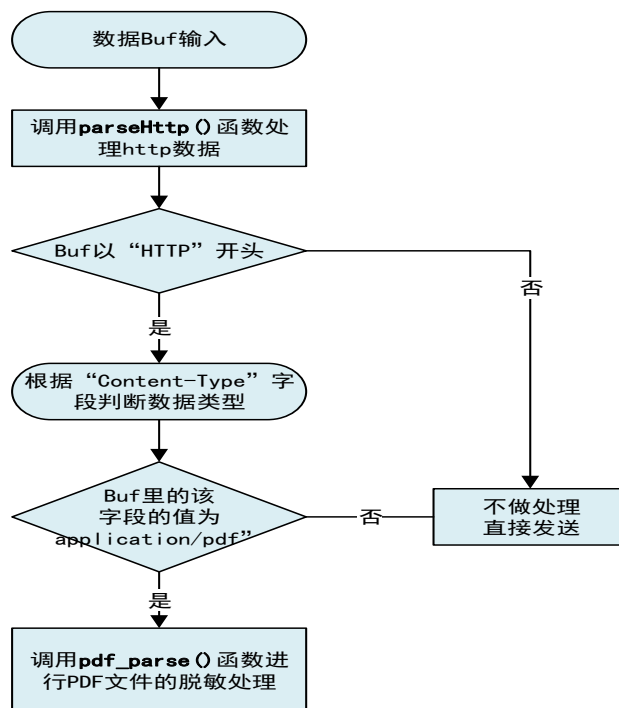


图 5-11 HTTP 解析流程示意图

### 5.3.3 内容解析模块

该模块的实现我们在第 3 章已经详解，这里我们不在赘述，主要对收到数据包进行内容解析，即对 stream 流进行解析，再进行脱敏目标的确认（敏感词的模式匹配）、以及脱敏策略的制定（脱敏算法）如图 5-12。该模块建立在 HTTP 协议解析模块基础之上，对发送和接收 HTTP 数据进行处理。

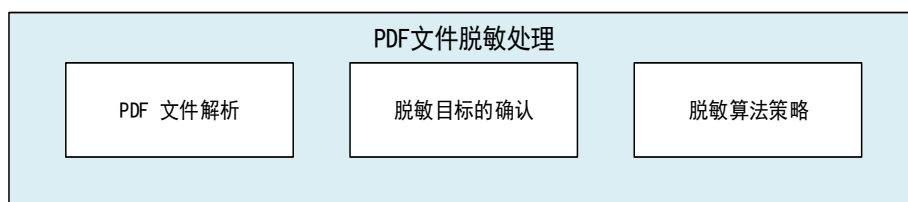


图 5-12 内容解析模块

其接口函数是 Parsepdf（），其具体形式 Parsepdf（char\* src,long file\_len, const vector<string> &s），参数 src：需要脱敏的 PDF 文档在内存中的起始地址，file\_len:PDF 文件大小。

### 5.3.4 敏感词控制模块

该模块是为了方便设定敏感词的操作，其图形界面的操作通过套接字通信实现，其通信过程如图 5-13：

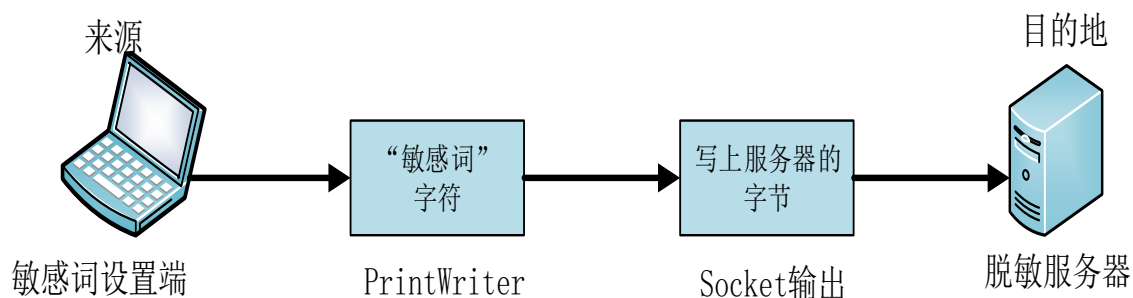


图 5-13 敏感词界面与服务器通信过程图

- 1) 对服务器建立 Socket 连接，{new Socket(serveIp,port)};
- 2) 建立链接到 Socket 的 PrintWriter，{new Printwriter(socket.getOutputStream())}，PrintWriter 是字符数据和字节间的转换桥梁，可以衔接 string 和 Socket 两端;
- 3) 写入数据，writer.println("textArea.getText()"); 点击按钮 Submit 触发发送，将发送缓存队列内容发送至脱敏服务器。

该界面如图 5-14，主要包括以下功能：



图 5-14 敏感词设置界面图

- 1) 输入(Insert, alt+i): 把脱敏关键字插入发送缓存队列，等待下一步操作;
- 2) 删除>Delete, alt+d): 删除已经存在于发送缓存队列中的脱敏关键字;
- 3) 提交>Submit, alt+s): 向脱敏服务器发送脱敏关键字;
- 4) 清空>Clear, alt+c): 清空发送缓存队列，等待新的脱敏关键字的输入;
- 5) 连接>connect, alt+r): 和脱敏服务器为 IP Address 建立连接。

以上几个功能按钮实现了第三方主机与脱敏服务器之间的通信交互，由于 windows 系统普遍使用 GBK 编码，而 linux 系统普遍使用 UTF-8 编码，故编译好代码后强制使

用 UTF-8 编码实现图形界面的实现，保证编码格式的一致性，实现编译命令：

```
javaw -Dfile.encoding=UTF-8 -jar form.jar
```

## 5.4 实验结果分析

### 5.4.1 实验环境搭建

本文除了对 PDF 文件进行敏感词脱敏处理以外，还对网络中基于 HTTP 协议的 PDF 文件，对其进行实时的敏感信息处理。测试环境由 2 台 PC 机（一台为用户、另一台为脱敏服务器），PC 机上安装有浏览器、虚拟机。

脱敏服务器的配置为：ubuntu 操作系统，内核版本为 16.04，配置为 Intel (R) Core (TM) i5-3230M CPU @2.60GhzX4 内存为 3.7GB，硬盘为 95.7G。

环境搭建如下：

- 1) 首先，用 ifconfig 命令查看脱敏服务器的 IP 地址,如：192.168.1.108，用文件 maskserver 记录，内容为 192.168.1.108 5500，5500 为端口号，一般为注册端口，范围为 1025-49151。
- 2) 用文件 Severhost 记录需被代理服务器（内容服务器）的 IP 地址，如 202.115.65.49 80、202.115.71.130 80；
- 3) 由于做解析时需要用到 zlib 库，因此需在系统下先安装好 zlib 库；
- 4) 编写 shell 脚本程序 strat.sh 用于启动脱敏服务器，并在 shell 脚本中写明了代理的服务端 IP 地址（80 端口），脚本代码如下：

```
#!/bin/bash
export LD_LIBRARY_PATH=./lib //检查环境变量是否配置好
IPADDR=`cat MaskServer` //读出脱敏服务器的 IP 地址
./bin/TuoMing_server $IPADDR //启动脱敏代理服务器
```

在启动脱敏代理服务器后，通过界面设置敏感词，凡是来自被代理内容服务器的 PDF 文件，其敏感词都可以成功脱敏。

### 5.4.2 脱敏效果测试

脱敏数据为网络中的 PDF 文件，从网络中下载 PDF 文件经过脱敏服务器把敏感词脱敏掉，包括正文文本内容和注释文本内容。经多次测试，都能较好完成脱敏工作。选择西南交通大学网站上面的 PDF 文件为例，网络中原 PDF 文件如图 5-15。

# 西南交通大学本科生转专业实施办法

(2016 年 1 月修订, 2016 年 3 月 2 次修订)

## 第一章 总则

第一条 根据教育部《普通高等学校学生管理规定》、《西南交通大学本科生学籍管理规定》，为进一步规范本科生转专业管理工作，便于学生合理安排学习计划，结合我校本科教学运行实际情况，特制定本实施办法。

第二条 本办法适用于我校按照国家规定录取的统考统招、接受全日制普通高等学历教育本科学生的转专业管理。

第三条 本办法不适用于中外合作办学机构和中外合作办学项目本科学生。

图 5-15 原 PDF 文件

假设设置敏感词为“本科”和“西南”，脱敏后效果如图 5-16。

# 交通大学 生转专业实施办法

(2016 年 1 月修订, 2016 年 3 月 2 次修订)

## 第一章 总则

第一条 根据教育部《普通高等学校学生管理规定》、《交通大学 生学籍管理规定》，为进一步规范 生转专业管理工作，便于学生合理安排学习计划，结合我校 教学运行实际情况，特制定本实施办法。

第二条 本办法适用于我校按照国家规定录取的统考统招、接受全日制普通高等学历教育 学生的转专业管理。

第三条 本办法不适用于中外合作办学机构和中外合作办学项目 学生。

图 5-16 经脱敏的 PDF 文件

由于在网页上找有注释的 PDF 文件相对不容易,因此为了展示注释的脱敏效果图,人为对相应的 PDF 文件加上了注释,再进行 PDF 文件脱敏操作。含注释的 PDF 原文件内容如图 5-17。



图 5-17 含注释的原 PDF 文件

假设设置敏感词为“法轮功”,脱敏后的效果如图 5-18。



图 5-18 注释脱敏后的 PDF 文件

从以上测试效果图,看出在不更改文件的格式的条件下,本系统能实时对经 HTTP 协议传输的 PDF 文件进行脱敏操作。

### 5.4.3 系统性能分析

本文通过测试 CPU 和内存占用情况分析该系统的性能,本实验分别对一次请求和并发多次请求做测试,一次和多次请求用压力测试工具 http\_load,其命令格式如下:

http\_load -p 并发进程数 -f 访问的总次数 需访问的 url

另外使用 top 命令看其内存和 CPU 占用情况,其格式如下:

top -p 进程号 -b -n 循环显示的次数 -d 屏幕刷新的时间 >写入文件的名字&

一次请求的 CPU 和内存占用情况如图 5-19。



图 5-19 一次请求 CPU 及内存占用

从图中，我们可以看出其 CPU 占用率和内存占用比较少，刚开始请求数据是开始占用 CPU，到对数据响应及脱敏服务时，CPU 占用稍微上升，经测试不经脱敏服务器的一次请求响应的时间为 0.44476s，经过脱敏服务器的时间为 1.03712s，这点时间差对用户的体验来说不会有很大的影响。

为测试在并发性情况下，CPU 及内存占用情况，分别针对最大并发数为 10、20、30、40、50 进行测试，测试结果如图 5-20。

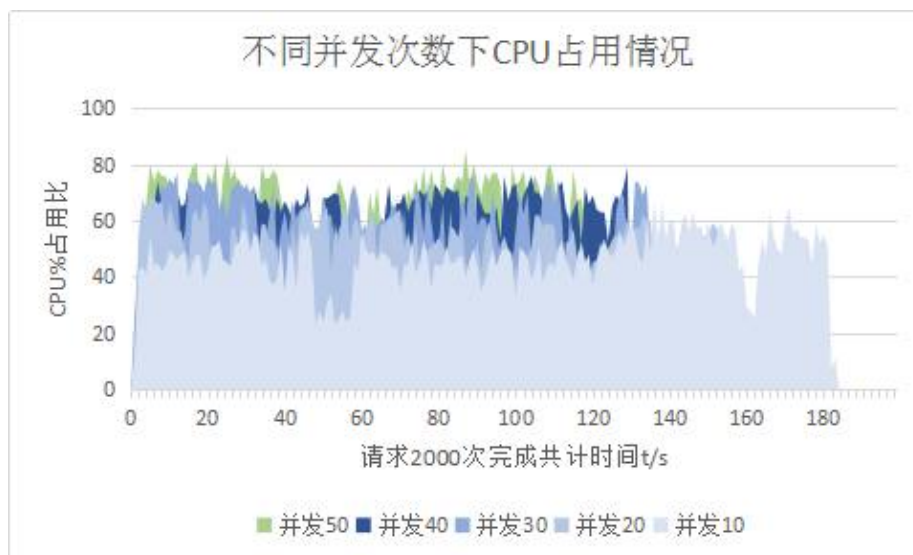


图 5-20 不同并发次数下的 CPU 占用情况

从图中可以看出，虽然随并发次数的增加，CPU 占用率增大，但情况还是比较良好；此外，不同并发次数、一次请求的耗时相差不大，说明并发情况对处理的速率影响比较小。

此外，为测其系统稳定性，连续三个小时内发起请求，并发进程为 20，其命令为：  
`http_load -p 并发进程数 -s 需访问的时间 需访问的 url`。

其 CPU 和内存占用情况如图 5-21。

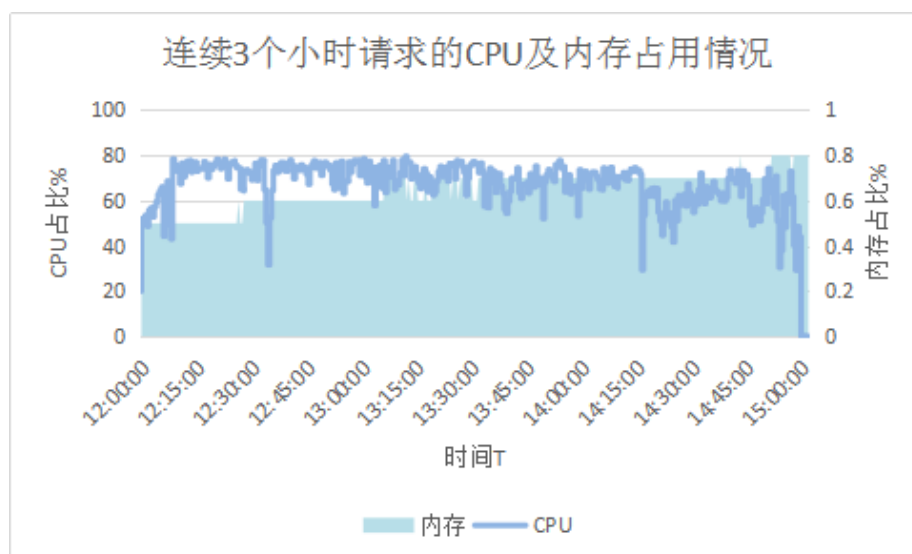


图 5-21 连续 3 小时请求 CPU 及内存占用情况

通过测试，在 3 个小时内，请求成功共 172317 个连接，230 次连接请求失败，从图中 CPU 及内存占用情况，该脱敏系统还是比较稳定。

## 5.5 本章小结

在深入研究 PDF 文件格式的基础上，为满足对网络中 PDF 文件在线脱敏需求，在基于代理的脱敏系统的框架，对 PDF 文件进行实时在线脱敏。本章通过在实验室搭建环境，通过代理机制，实现对网络中的 PDF 文件脱敏的需求，并验证其系统的功能性和稳定性都比较良好。

## 结 论

本章将对本论文的工作进行总结，并对后续进一步的研究问题进行讨论。

### 论文工作总结

随着互联网技术的大力发展、大数据时代的登场，对通信网络的信息安全的要求也越来越高，这不仅是国家所重视的，也是我们每个人所关注的，担心自己的隐私信息被泄露。因此在保证资源共享性、开放性的前提下，防止涉密信息泄露成为一大关键问题。

本文在强调通信网络信息安全的重要性的基础上，对当前使用范围最广的 PDF 文档进行研究，通过看标准和大量文献，了解其研究现状，总结了 PDF 文件格式，并在前人解析 PDF 文件的方法的基础上，提出了一种基于 Stream 流的 PDF 解析方法；与前人的方法不同在于，无需考虑逻辑性和处理大量的对象，只对我们所需的文本对象、Cmap 流对象进行处理，不仅降低了逻辑复杂度，还能对网络的中部分的 PDF 文件进行实时在线解析。

同时为达到高效脱敏，在敏感目标确认上入手，在 BM 和 QS 算法的优点上，利用 PDF 文件的编码格式，提出一种适用于 PDF 文件地文本内容高效匹配算法，通过实验验证，该算法具有一定地高效性。

最后再基于代理机制的脱敏系统下，搭建环境，实现对网络中 PDF 文件的脱敏处理，并验证了其功能良好性和系统的稳定性。

### 未来工作展望

在下一步的研究和学习过程中，将在以下方面做出改进：

1) PDF 文件编码格式太复杂，本文只能对大部分 PDF 文件进行解析，针对某些特殊的 PDF 文件，并且本文只针对文本内容，可能还会存在图片、视频等多种形式，还有待研究。

2) 本文提出的改进算法，目前只试用于 PDF 文本内容的匹配，可是在进行脱敏的过程中，还需要进行其他关键词的匹配，后期还需在此基础上改进算法，能多覆盖各种匹配，以此提高性能。

3) 针对网络中 PDF 文件，本文可实现在线处理，部分 PDF 文件可实现实时处理，目后期需研究并改进满足都能做实时处理。

4) 由于本文是基于 Stream 流操作的，由于 PDF 中大部分是 Stream 对象，这样并行化处理流内容就是可行了，后期可考虑，以此提高处理速度。



---

5) 此外, 本文针对 PDF 文件涉密信息的保密性的应用, 后续可对敏感信息脱敏次数, 进行舆情监控等应用。

---

## 致 谢

3 年前带着憧憬与忐忑来到交大，时光如箭，转眼间，3 年的研究生学习和生活即将结束。在交大的这 3 年对我人生有着比较深的影响，令我难以忘怀。在此，向一路走来为我提供帮助的人致以诚挚的谢意。

首先，我要感谢我的导师陈庆春教授，记得陈老师说过不放弃，会陪同学们一起走到最后。整个研究生期间，陈老师不仅给予学习上的指导，还给予生活上无私的关怀与帮助。陈老师渊博的专业知识、严谨的学术态度和兢兢业业的工作态度给我留下深刻的印象。感谢陈老师在我刚进校的时候，就让我参与项目，在项目组中，我的沟通能力、分析问题及解决问题的能力得到一定的提高，让有着项目经验的我找到一份比较满意的工作；此外在陈老师的悉心指导下，基于项目，完成了毕业论文，感谢陈老师给予的指导及宝贵的意见。

其次，感谢师兄兰小龙、刘勇博士和超有耐心的银帆博士，在项目期间提供给我学业上的帮助；感谢师弟周国剑、刘邦国、骆德腾在写论文期间给我提供的意见与帮助，感谢师弟王旌舟和我一起并肩作战 PDF；感谢廖怨婷、辛宏涵、钱巧娅、杨柳、牟桐、盛楠、师妹马云琪、王闻祎、师弟肖观腾，你们在研究生期间，给我生活上带来的欢乐，这将成为我人生最美好的回忆。

感谢我的朋友们，“段子手”饶齐、“戏精”周俊梅、“呆萌”熊露、“花姐”张华、“大腿”钟师兄和“川胖子”张海川这 3 年来互相鼓励、给予精神上的支持以及给我带来的欢声笑语。

还要感谢我的父母，给予我生命，育我成长，在 20 多年的求学道路上，一直做我坚强的后盾，与我同在；当我不开心的事，他们会耐心的听我倾诉；当面临人生重要的抉择时，他们会给我分析利与弊，最后义无反顾支持我的决定，对我的恩情无以回报。感谢一路以来你们对我无微不至的关怀！

## 参考文献

- [1] 第 41 次《中国互联网络发展状况统计报告》发布[J]. 中国广播, 2018(03): 96.
- [2] 《信息通信网络与信息安全规划(2016-2020)》正式发布[J]. 现代电信科技, 2017, 47(01): 77.
- [3] 张梦. 通讯网络信息安全防护的强化[J]. 电子技术与软件工程, 2017(17): 206-206.
- [4] 高阿朋, 孙晓达, 刘卫明, 等. 试论关于通信网络信息安全防护[J]. 电子测试, 2017(22).
- [5] 李元峰. 涉密信息系统保密技术检查探索体会[J]. 信息安全与通信保密, 2013(6): 74-76.
- [6] 周海涛. 大数据平台数据脱敏关键技术[J]. 电子技术与软件工程, 2017(21): 150.
- [7] 谷德丽, 马春光, 郭金生, 等. 基于可信连接的涉密信息系统安全防护[J]. 信息安全与通信保密, 2012(12): 132-134.
- [8] 江堂碧. 支持挖掘的流式数据脱敏关键技术研究[D]. 电子科技大学, 2017.
- [9] 刘明辉, 张尼, 张云勇, 等. 云环境下的敏感数据保护技术研究[J]. 电信科学, 2014, 30(11): 1-8.
- [10] Cost of Data Breach: Global Analysis [EB/OL]. Ponemon Institute, 2013.
- [11] Sarada G, Abitha N, Manikandan G, et al. A few new approaches for data masking[C]// Proceedings of the International Conference on Circuit, Power and Computing Technologies. IEEE, 2015: 1-4.
- [12] Vijayarani S, Tamilarasi A. An efficient masking technique for sensitive data protection[C]// Proceedings of the International Conference on Recent Trends in Information Technology. IEEE, 2011: 1245-1249.
- [13] Li M, Liu Z, Jia C, et al. Data Masking Generic Model[C]// Proceedings of the Fourth International Conference on Emerging Intelligent Data and Web Technologies. IEEE Computer Society, 2013: 724-727.
- [14] 卞超轶, 朱少敏, 周涛. 一种基于保形加密的大数据脱敏系统实现及评估[J]. 电信科学, 2017, 33(3): 119-125.
- [15] 马然. 基于深度学习的自然场景文本识别系统的设计与实现[D]. 吉林大学, 2015.
- [16] 陈天莹, 陈剑锋. 大数据环境下的智能数据脱敏系统[J]. 通信技术, 2016, 49(7): 915-922.
- [17] 刘平. ScienceWord 软件中 PDF 文档的生成及其内容提取研究[D]. 华中科技大学, 2007.
- [18] 牛永洁, 薛苏琴. 基于 PDFBox 抽取学术论文信息的实现[J]. 计算机技术与发展,

- 2014(12): 61-63.
- [19] 张秀秀, 张立峰. PDF 文件文本内容提取研究[J]. 图书情报导刊, 2008, 18(36): 118-120.
- [20] 李珍, 田学东. PDF 文件信息的抽取与分析[J]. 计算机应用, 2003, 23(12): 145-147.
- [21] 李贵林, 李建中, 杨艳. 用 Plug-in 实现对 PDF 文件的信息提取[J]. 计算机应用, 2003, 23(2): 110-112.
- [22] 宋艳娟. 基于 XML 的 HTML 和 PDF 信息抽取技术的研究[D]. 福州大学, 2006.
- [23] 范洪博. 高性能精确单模式串匹配算法研究[D]. 哈尔滨工程大学, 2009.
- [24] Yao C C. The complexity of pattern matching for a random string[J]. Siam Journal on Computing, 2006, 8(3): 368-387.
- [25] Boyer R S, Moore J S. A fast string searching algorithm[M]. ACM, 1977.
- [26] J. H. Morris, Jr and V. R. Pratt. A linear pattern-matching algorithm[R]. University of California, Berkeley, 1970.
- [27] Knuth D E, Jr J H M, Pratt V R. Fast Pattern Matching in Strings[J]. Siam Journal on Computing, 1977, 6(2): 323-350.
- [28] Horspool R N. Practical fast searching in strings[J]. Software Practice and Experience, 1980, 10(6):501-506.
- [29] Franek F, Jennings C G, Smyth W F. A simple fast hybrid pattern-matching algorithm[J]. Journal of Discrete Algorithms, 2007, 5(4): 682-695.
- [30] Sunday, Daniel M. A very fast substring search algorithm[J]. Communications of the Acm, 1990, 33(8): 132-142.
- [31] Journal D. Fast String Searching[J]. Software Practice and Experience, 2010, 21(11): 1221-1248.
- [32] Berry T, Ravindran S. A Fast String Matching Algorithm and Experimental Results[C]// Proceedings of the Stringology. 1999: 16-28.
- [33] Sheik S S, Aggarwal S K, Poddar A, et al. A fast pattern matching algorithm[J]. Journal of chemical information and computer sciences, 2004, 44(4): 1251-1256.
- [34] Thathoo R, Virmani A, Lakshmi S S, et al. TVSBS: A fast exact pattern matching algorithm for biological sequences[J]. J.indian Acad.sci.current Sci, 2006, 91(1): 47-53.
- [35] 刘泷, 高仲合, 宋先强,等. Boyer-Moore 模式匹配算法的一种改进算法[J]. 电子技术, 2016, 45(11).
- [36] 杨旭阳. 卫星通信网络入侵检测技术研究[D]. 河北科技大学, 2016.
- [37] 赵晓, 何立风, 王鑫,等. 一种高效的模式串匹配算法[J]. 陕西科技大学学报, 2017,

- 35(1): 183-187.
- [38] 徐政超. 模式匹配 BM 算法改进探讨[J]. 科技与创新, 2017(6):59-59.
- [39] 于波涛. 基于文档属性的 PDF 数学表达式信息获取[D]. 河北大学, 2015.
- [40] Inc A S.PDF Reference: Version 1.7[Z]. 2006.
- [41] Adobe Systems Inc.Adobe Technote 5014:Adobe CMap and CIDFont Files Specification,Version1.0[EB/OL]. Adobe Systems Inc.[http://partners.adobe.com/public/developer/en/font/5014.CIDFont\\_Spec.pdf](http://partners.adobe.com/public/developer/en/font/5014.CIDFont_Spec.pdf).
- [42] 刘丽荣. 格式化文件内容提取与过滤关键技术研究[D]. 哈尔滨工程大学, 2012.
- [43] Adobe Systems Inc.Adobe Technote 5411: ToUnicode Mapping Files Specification, 2003[EB/OL].Adobe SystemsInc.<https://www.adobe.com/content/dam/acom/en/devnet/acrobat/pdfs/5411.ToUnicode.pdf>.
- [44] Ravikumar G K, Manjunath T N, Hegadi R S, et al. A Survey on Recent Trends, Process and Development in Data Masking for Testing[J]. International Journal of Computer Science Issues, 2011, 8(2): 1709-20.
- [45] 缪旭东, 王永春, 曹星辰,等. 基于模式匹配的安全漏洞检测方法[J]. 计算机科学, 2017, 44(4): 109-113.
- [46] 徐周波, 张永超, 古天龙,等. 面向入侵检测系统的模式匹配算法研究[J]. 计算机科学, 2017, 44(9): 125-130.
- [47] 王樱, 杨丽, 李锡辉. 模式匹配技术在多序列比对中的应用[J]. 信息系统工程, 2014(10): 79-81.
- [48] 郝春媚. 涉密检查系统中信息检索技术研究[D]. 北京邮电大学, 2014.
- [49] Aho A V,Corasick M J.Efficient String Matching:An Aid to Bibliographic Search[J]. Communications of the Acm, 1975, 18(06): 333-340.
- [50] Wu S,Manber U.A Fast Algorithm for Multi-pattern Searching[Z]. Report TR-94-17, Department of Computer Science, University of Arizona, Tucson, AZ, 1994.
- [51] 徐成, 孙伟, 戴争辉,等. 一种面向入侵检测的 BM 模式匹配改进算法[J]. 计算机应用研究, 2006, 23(11): 89-91.
- [52] 杨薇薇, 廖翔. 一种改进的 BM 模式匹配算法[J]. 计算机应用, 2006, 26(2): 318-0319.
- [53] 蒋亚平, 田月霞, 赵军伟. 一种改进的 BM 模式匹配算法[J]. 科技通报, 2015(9): 178-182.
- [54] 曾传璜, 段智宏. 一种改进的 QS 串匹配算法[J]. 计算机与数字工程, 2010, 38(7): 48-49.
- [55] 巫喜红, WUXihong. 改进的 QS 模式匹配算法的性能分析[J]. 计算机工程与应用,

---

2014, 50(2): 44-48.

[56] 晓涵. HTTP 协议揭秘[J]. 计算机与网络, 2017, 43(Z1): 64-71.

[57] 王晔. 基于 HTTP 反向代理的 Web 安全网关技术研究[D]. 扬州大学, 2016.

[58] Fielding R, J. G S, Mogul J, et al. RFC 2616: Hypertext Transfer Protocol - HTTP/1.1[J]. Computer Science & Communications Dictionary, 1999, 7(9): 3969 - 3973.

[59] DavidGourley, 古尔利, Totty,等. HTTP 权威指南[J]. 2012.

---

## 攻读硕士期间发表的论文及科研成果

已发表的论文:

- [1] 朱玲玉, 王旌舟, 陈庆春. 适用于 PDF 文本内容的高效模式匹配算法[J]. 通信技术, 2018, 51(03): 641-646.



知网查重限时 7折 最高可优惠 120元

本科定稿，硕博定稿，查重结果与学校一致

立即检测

免费论文查重: <http://www.paperyy.com>

3亿免费文献下载: <http://www.ixueshu.com>

超值论文自动降重: [http://www.paperyy.com/reduce\\_repetition](http://www.paperyy.com/reduce_repetition)

PPT免费模版下载: <http://ppt.ixueshu.com>

---