本科毕业设计（论文）

影视导航网站影视数据处理策略的设计与实现

The design and implementation of the data processing strategy of the film and television navigation website

学　　院：软件学院

专　　业：软件工程

学生姓名：×××

学　　号：12301065

指导教师：×××

北京交通大学

2016 年 5 月

学士论文版权使用授权书

本学士论文作者完全了解北京交通大学有关保留、使用学士论文的规定。特授权北京交通大学可以将学士论文的全部或部分内容编入有关数据库进行检索，提供阅览服务，并采用影印、缩印或扫描等复制手段保存、汇编以供查阅和借阅。

（保密的学位论文在解密后适用本授权说明）

学位论文作者签名：　　指导教师签名：

签字日期：　　年　　月　　日　　签字日期：　　年　　月　　日

中文摘要

摘要：随着现代信息技术的飞速发展，互联网已经慢慢渗透到人们生活工作的各个方面。网址导航作为一个集合多个网站的网址，并按照一定条件进行分类的网址站，它能够方便网友们快速找到自己需要的网站，而不用去记住各类网站的网址，直接进到所需的网站。

影视导航网站是众多导航网站中的一个。由于网站的一系列视频数据都是来源于其他网站，如搜狗影视导航网站目前站点来源有：搜狐视频、乐视、优酷、土豆、迅雷、腾讯视频等。但是这些网站对同一部影片的标题、简介、内容描述等都存在着差异，如果在入库之前不加以分类整合，长期积累的无效数据会给数据库带来很大的压力，为了解决这个问题，我们需要在获取数据之后先设计数据处理策略将这些数据按制定的规则做归一化处理（主要是整合标题简介等内容、判定可入库和不可入库的数据等）再入库，之后再进行分类、排序等一系列操作，最后将这些视频数据反应到前端。

本文的目标是根据对入库所需数据的最终结果和技术途径的分析，提出数据处理的策略，并根据这个策略处理抓取的 XML 数据，使前台反映出的数据尽最大可能有效化，方便用户体验。

目前，该导航网站已成功上线，点击量比较高，给公司带来了可观的收益。

关键词：导航网站；信息抓取；数据归一处理

ABSTRACT

ABSTRACT:

With the rapid development of modern information technology, the Internet has penetrated into every aspect of people's life and work. Site navigation as a set the URL of the website, and in accordance with the conditions of classification of the web station. It can help users to quickly find their site, without having to remember various sites on the web site, directly into the desired site.

Movie and TV navigation website is one of the many navigation websites. Because a series of video data website are from other sites, such as the site of the current source of Sogou Video navigation site: Sohu video, music, Youku, potatoes, thunder, Tencent, etc.. But the title of a film of these sites, introduction, content description, there is a difference, if before storage can not be integrated classification, long-term accumulation of invalid data will bring great pressure to the database, in order to solve this problem, we need to access to the data in the first design data processing strategy will be the data according to the rules of the normalized (mainly is to integrate the title and introduction, determine the storage and not be stored data, etc.) and then put in storage, and then classification, sorting, and so on a series of operations, in the end, the video data of the response to the front.

The goal of this paper is according to the analysis of the data on the storage required final results and technical approaches proposed data processing method, and grab the XML data processing according to the strategy, and to make the front desk reflects the data to the greatest extent possible effective, convenient user experience.

At present, the navigation site has been successful on-line, click on a relatively high, to the company has brought considerable benefits.

KEYWORDS： Site navigation; information extraction; data normalization processing

目　　录

1 引言

本章为引言部分，主要是介绍项目的背景和意义，简述开发中的重难点以及笔者在开发中参与的工作，并简单阐述本文的组织结构。

1.1 项目的背景及意义

随着当代信息技术的飞速发展，互联网已经慢慢的渗透到人们生活工作的各个方面。它是一个以高速传播为长处的平台。在满足的了人们的共性需求之后，就诞生了具有领域特色的、垂直专注某一领域的网站来满足人们更多的需求。

网址导航（Directindustry Web Guide）是一个按照一定的条件进行分类，集合众多网址的网址站，它是互联网最早的网站形式之一。网址导航不需要网友记住各类网站的网址，而是直接整合，方便网友快速找到自己需要的网站并直接进入。正因为网址导航的模式简单、服务便利、用户体验极好的特点，获得了极大好评，但是这让它的发展和竞争了成为互联网网站中竞争最激烈的类别。

影视导航网站是众多导航网站中的一种。影视导航网站的站点来源是以各 xml 中 webName 和 webSiteUrl 为指定站点，和已登记的站点进行对应，不在登记站点列表内的 webname 或 websiteurl 则 xml 无法入库。搜狗影视导航网站目前站点来源有：搜狐视频、奇艺、乐视、优酷、土豆、迅雷、电影网、PPTV、腾讯视频、酷米、华数；其中电影网只提供电影和综艺，华数只提供综艺，酷米只提供动漫。但是这些网站对同一部影片的标题、简介、内容描述等不尽相同，在获取数据之后就需要先将这些数据按一定的规则做归一化处理再入库，即合并标题、简介等内容，筛选入库的数据，更新、修改、删除来自各个网站的视频数据，根据豆瓣影评对应到相应的视频中等一系列工作，以便于用户快速查找，在短时间内找到最合适的影片资源，同时，也能拥有自己网站的特色，提高竞争力，同时避免数据重复，减轻数据库压力。

1.2 项目的重点与难点

该项目的难点主要如下：

（1）影视数据信息量大，如何精准的分析和归类。

（2）归并策略的设计：制定从各个视频网站获得的影视数据的归并和不归并的规则。

（3）在保障网站性能和减轻服务器压力的前提下，完成数据归一化处理提高数据库效率及减轻数据库压力。

1.3 作者的主要工作

作者参与整个项目的开发过程，并在项目上线后负责对系统进行维护，开发阶段的主要工作为：

（1） 参与整个系统的架构设计。

（2） 数据抓取模块的开发。

（3） 入库前数据处理策略的设计，包括可入库数据和不可入库数据的要求，数据归一处理的规则设计。

（4） 数据归一模块的开发。

（5） 系统测试，主要测试是否成功实现了数据归一的功能。

1.4 论文的组织结构

本论文共分为七章，具体内容安排如下：

第一章介绍了项目的背景与意义，简单说明了笔者在本毕业设计中负责的主要工作，并且介绍了论文的基本组织结构。

第二章介绍了毕业设计中主要设计到的的相关技术，包括 PHP、正则表达式、Mongodb 数据库、Redis 数据库、Nginx 服务器、Mcpack 工具库和单元测试。

第三章介绍了本影视导航网站数据处理部分的功能性需求和非功能性需求。并对数据处理的各个模块进行了说明。

第四章介绍了系统的概要设计。主要包括数据整合处理策略的设计，数据处理系统的架构设计和数据处理策略的设计。

第五章介绍了系统的详细设计与实现。主要包括数据抓取模块、数据更新与修改模块、数据删除模块和数据归并排序模块。通过一系列的图表展示笔者的设计理念。

第六章通过对测试方法的介绍，结合测试用例，说明了笔者是如何对系统进行功能性测试和性能测试的，并对测试结果进行了分析。

第七章是总结，说明了系统的不足，说出在未来可行的发展方向，对系统的发展前景做出了展望。

1.5 本章小结

本章简单的介绍了这个项目的背景和意义，也介绍了项目的重难点，同时也介绍了笔者在项目中担任的主要工作，并且对论文的组织结构进行详细的分析。

2 相关技术介绍

本章主要介绍笔者在毕业设计中用到的一系列技术，包括 PHP，用于处理字符串的正则表达式，Mongodb 数据库、Redis 数据库，接口数据打包工具 Mcpack 和单元测试相关。

2.1 PHP 概述

PHP 是一种开源的通用计算机脚本语言，是 HTML 内嵌式的语言，它的语法混合了 C 语言，Java，Perl 和 PHP 式的新语法，利于程序员的学习。尽管它混合了多种语言，但是和 Perl 相比，由于它是直接把程序嵌入到 HTML 文档中执行，这就使得它能够在更短的时间里快速的执行动态网页，不需要很多知识就能建立一个交互的 web 站点来管理动态的内容、处理会话跟踪等。它支持当下许多流行的数据库，包括 MySQL、oracle、Microsoft SQL server 等，这也使得它更能成为人们的第一选择[1]。

PHP 主要是 web 环境下，以客户端浏览器为载体，传递需要的信息给 web 服务器，然后由 web 服务器启动事先指定的程序码来完成特定的工作[2]。正因为它的脚本是在服务器上，并非浏览器上运行，而服务器上的脚本不易被复制，因为用户只能看到解释后的结果，看不到浏览页上的 PHP 源代码，这就保障了源程序的保密性。

2.2 正则表达式概述

正则表达式是一种处理字符串的工具，其文本模式包括普通字符（如：a 到 z 之间的字母）与特殊的字符（也称之为“元字符”）。它用已定义好的一些特殊字符及其组合组成一个规则字符串用以表达对字符串的过滤逻辑，从而实现替换、匹配、提取字符串的功能[3]。

得益于它的灵活性、逻辑性和功能性强的这些特点，正则表达式能用快速简便的方式实现字符串的复杂控制，包括判断所给的字符串是否符合正则表达式的过滤逻辑、从字符串中获取我们需要的部分等功能，因此，它在处理文本内容方面的应用十分广泛。

2.3 Mongodb 概述

Mongodb,即分布式文档存储数据库，由C++编写，主要是为web系统应用提供可扩展的高性能数据存储解决方案。它支持的是数据结构松散类似于 json 的bson格式，能够存储较为复杂的数据类型。

在数据量超过50G之后，MongoDB的访问速度比MySQL快十倍有余。不过MongoDB的并发读写效率并不是非常出色，官方提供的性能测试里的数据显示，它每秒可处理大约0.5万~1.5万次读写请求。MongoDB还自带了一个名为GridFS的分布式文件系统，用以支持海量的数据存储[4]。

Mongodb最大的特点是它可以支持非常强大的查询语言，语法类似于面向对象的查询语言，可实现类似关系数据库单表查询的大部分功能，同时，它还支持对数据建立索引。高性能、易部署、易使用这些特点使得用它来存储数据非常的方便。

2.4 Redis概述

Redis是一个基于内存可持久化的日志型key-value数据库，它还提供了多种语言的API。它是用开源的使用ANSI C语言编写的，支持的value类型比Memcached更多，包括了string，list，set，zset，和hash。Redis支持各种不同的排序方式，把数据缓存在内存里，周期性的把更新的数据写进磁盘或把修改操作写进追加的记录文件中，以此为基础，实现了主从同步。数据是在主服务器上向任意数量的从服务器上同步，这其中的从服务器也可以是关联了其他服务器的主服务器，这个特点也就方便了单层树复制的执行。

2.5 Mcpack

Mcpack是一个用于模块间接口数据打包的工具库。它通过网络传输，提供函数助于组织接口数据，把数据打包成二进制的包来传输。其优点在于能够方便简洁的实现向下兼容，因为它具有十分丰富的内建数据类型，让它可以在文本格式和二进制格式之间自动进行相互转换。用mcpack作为接口还可以提高程序的可配置性和可测试性。

Mcpack作为一种内存对象的序列号形式，能够把内存中的数据结构序列化为与计算机架构无关的bit流，再由bit流重建对象，使所有产品的接口趋于统一。

2.6 单元测试

单元测试是软件测试粒度最小的模块，面向对象中就是针对方法的测试。集成测试测试基于单测，介于系统和单元测试之间、模块之间的核心业务测试。

单元测试衡量的标准为覆盖率（语句覆盖、条件覆盖、判定覆盖、路径覆盖）、准确率（case 通过率）、健壮性、可执行性。此次项目中其设计原则为：

（1）目标明确：熟悉功能需求；

（2）熟悉业务框架：了解功能实现过程；

（3）代码覆盖率：重视覆盖率，但不需要一味的追求；

（4）Case 命名规范：以 test 开头，表名测试方法名，且方法名首字母大写

（5）注释规范：注释清晰、易读，每条用例都标明所测的功能

（6）边界值覆盖

（7）Case 独立性

（8）外部依赖隔离

（9）单一原则

（10）执行时间短

（11）自动化

常用的测试工具/框架主要是：testng，junit，jmockit，springtest，feed4junit 等。

2.7 本章小结

本章主要介绍了该系统在开发与测试过程中使用的关键技术。系统主要使用 PHP 语言开发，在开发和测试中用到了 Mongodb、Redis、Mcpack、正则表达式等技术，本章对这些技术的基本概念和特点等方面进行了简要的概述。

3 系统的需求分析

本章通过功能性需求分析和非功能性需求分析这两个部分来对系统的整体需求进行说明，将系统的各个模块进行详细的功能性需求分析，接着再对系统的非功能性需求进行分析。

3.1 功能性需求分析

导航网站对数据的处理主要分为下面几个模块：XML 数据抓取、数据更新与修改、数据删除、数据归并、默认播放的选取、豆瓣数据抓取、豆瓣数据匹配、数据排序、数据分类。笔者将根据这些模块的数据处理要求进行功能分析。

3.1.1XML 数据抓取

导航网站由于只是提供一个整合数据的平台，其数据都是源于其他网站，站点来源为：各 xml 中 webName 和 webSiteUrl 为指定站点，和已登记的站点进行对应，不在登记站点列表内的 webname 或 websiteurl 则 xml 无法入库。webName 和 webSiteUrl 确定后，各合作方不可擅自变更，变更需要人工沟通。目前站点来源：搜狐视频、奇艺、乐视、优酷、土豆、迅雷、电影网、PPTV、腾讯视频、酷米、华数；其中电影网只提供电影和综艺，华数只提供综艺，酷米只提供动漫。

由于视频数据来源众多，但是其中无效或重复的数据过多，我们需要使用数据挖掘技术来提取有效数据，数据挖掘主要是根据数据挖掘任务需要的模式类型指定，其中数据挖掘任务分为描述和预测两类。在抓取数据时，将视频分为完结和持续更新两类，又根据视频类型的不同对抓取数据的要求进行了进一步的细分：

（1）电影 xml 分为全量和增量进行抓取，全量是指所有电影，增量是指新增电影，全量增量的字段内容一致。全量入库后每个月更新一次(用于查漏补缺)，电影增量每天更新若干次（20 分钟更新一次）。

（2）电视剧 xml 分为全量和增量进行抓取，全量是指已完结电视剧，增量是指更新中电视剧，全量增量的字段内容一致。全量入库后每个月更新一次（用于查漏补缺），电视剧增量每天更新若干次（5 分钟更新一次）。

（3）动漫 xml 分为全量和增量进行抓取，全量是指已完结动漫，增量是指更新中动漫，全量增量的字段内容一致。全量入库后每个月更新一次（用于查漏

补缺），动漫增量每天更新若干次（20 分钟更新一次）。

（4）综艺 xml 只有增量进行抓取，包括所有综艺节目。每天更新若干次（20 分钟更新一次）。

进行数据挖掘时要根据分类来构建一个模型，这个分类其实就是找出一个概念描述，用这个描述来代表这部分数据的整体信息，又或者称作内涵描述，用规则或决策树的模型来表述。而上述的抓取要求其实就是用粗糙集方法，从众多不确定的数据中利用这个软计算方法，通过分析类中对象共同特征（已完结/仍在更新中）来设计的，全量和增量之间不具有相同的特征，保证了获取数据的时候不会出现冲突。

3.1.2数据更新与修改

在 XML 入库之后，根据抓取数据时分的两类：全量和增量，其中全量是一月一更新，用于查漏补缺，通常是进行数据的修改，增量 20 分钟一更新，通常要对数据进行更新，基于 XML 数据抓取时使用的数据挖掘技术，在数据更新和修改时候的要求根据不同的视频类型依然进行了再次的划分：

（1）电影 xml 入库后，除中间页地址 listlink 外，剩余所有字段可修改（含 playlink），即字段不进行更新。Listlink 为唯一性标识。

（2）电视剧 xml 入库后，除中间页地址 listlink，剩余所有字段可修改（含分集 playlink），即字段不进行更新。Listlink 为唯一性标识。nowEpisode 可修改，但下次更新时即被覆盖。

（3）动漫 xml 入库后，除中间页地址 listlink 外，剩余所有字段可修改（含分集 playlink），即字段不进行更新。Listlink 为唯一性标识。nowEpisode 可修改，但下次更新时即被覆盖。

（4）综艺 xml 入库后，除中间页地址 listlink 和各个分期信息外，剩余所有字段可修改（含分期 playlink 和 curEpisode），即字段不进行更新。Listlink 为唯一性标识。

由于数据的更新和修改是在抓取数据的基础上，在此之前，视频已经根据完结与否进行了大体的划分，所以这个模块是利用正则表达式对字段进行处理。

3.1.3数据删除

数据的删除和更新与修改类似，删除接口统一为 del.xml，del.xml 需要进行改版升级。主要支持如下功能。

（1）电影删除支持 listlink 和 playlink，两者中一个均可使改电影标注为删除状态。删除后这个 url 不可重新入库，但管理员可以从后台恢复正常状态（可重新入库状态）。

（2）电视剧、动漫和综艺支持 listlink 删除，即整部删除，这种删除可使整部电视剧或动漫或综艺标注为删除状态。删除后这个 url 不可重新入库，但管理员可以从后台恢复正常状态（可重新入库状态）。

（3）只有一家播放站点资源的删除，播放按钮不显示，该影片不参与排序和搜索，但页面可以直接打开；多家播放站点资源的删除，删除掉对应的站点 icon 即可。

（4）电视剧、动漫和综艺支持单集（单期）删除，某个或多个 playlink 被标记为删除后，进入单集删除状态。单集删除后这个 url 可以重新入库，管理员也可人工恢复。

（5）删除的集数为最后的一集或多集，不影响排序展现。nowEpisode 正常减少为删除后最大集数。

（6）删除的集数为中间或前面的某集，影响了排序展现。nowEpisode 不变。空缺状态的集数展示方式如下，空缺集数显示“暂无 XX”，且不提供链接。例如：http://v.360.cn/tv/PrFtaK3oSGPnNX.html

（7）人工删除：整部删除和单集删除都支持人工删除。对于只有一个播放源的电影、电视剧、综艺、动漫，删除整体数据或删除单个播放源是同等效果，即删除其对应的 listlink。对于有多个播放源的电影、电视剧、综艺、动漫，删除整体数据即删除所有播放源对应的 listlink。单集删除为进入某个播放源的剧集列表，选择删除具体的某一期或某一集。

在数据恢复方面，整部删除建立回收站（包含接口删除的、人工删除；被删除的 listlink 的回收站），可以从回收站中选择复原，即 listlink 置为正常状态。单集删除建立回收站（包含接口删除的、人工删除；被删除的 playlink 的回收站），可人工恢复，即 playlink 成为正常状态。

3.1.4数据归并

来自多个视频站的数据，需要进行归一化，即合并为一条影片记录。归并在入库时进行，已经归并好的影片，下次更新和运算时继续参与和新数据的归并。各类型的视频在数据归并的规则方面也存在差异。大体是通过标题、年份、导演、主演、简介、地域等方面进行匹配。由于数据量大，因此在合并数据时是根据逐个视频类型进行的，以免造成视频数据的混乱，同时这个工作要在备份服务器上

进行，经测试无误之后，再把合并数据和未能匹配的数据导入正式库。

因需要归一的部分过多，这里以标题为例，对标题的处理规则如下：

（1）标题中的：冒号、·间隔号、！感叹号、括号和反括号均统一为半角，例如 “硬汉 2:奉陪到底”和 “硬汉 2：奉陪到底”

（2）标题中的字母大小写不区别，例如 “hello!树先生”和 “Hello！树先生”

（3）标题中的全角、半角空格均不进行区别，例如“无间道 之终极无间”和“无间道之终极无间”

（4）标题中末尾的Ⅰ、全角的 1 、半角的 1、半角的（一）、全角的（一）、第 1 部、第一部，直接去掉后参与匹配，例如“无间道 1”和“无间道”标题中末尾的ⅡⅢⅣ、全角的 2 3 4 、半角的 2 3 4、半角的（二）（三）（四）、全角的（二）（三）（四）、第 2 部第 3 部第 4 部、第二部第三部第四部，可以进行一一相互对应，例如 “高海拔之恋 2”和“高海拔之恋Ⅱ”

处理的方法则是利用正则表达式，用单个字符串描述、匹配符合上述规则的字符串，并替换成符合规定模式的文本，不过验证其语法是否正确的过程比较麻烦，需要一段段验证，在语句有包含、互斥等情况时，先取最小部分，然后再逐步扩大。

3.1.5默认播放的选取

默认播放在入库后选取好，即把其 playlink 排在第一位，剩余依次排列；其它信息类字段选择默认播放站点的相应字段内容（入库时已定好，不需更新），电视剧、动漫的全 XX 集和更新到 XX 集（nowEpisode）采用默认播放站点的，其中 nowEpisode 按照前文所述更新逻辑变化或更新。下次更新时，有新数据进入时仍然参与默认播放的选取运算。

默认播放可人工指定，人工指定后默认播放不再变化，剩余站点按策略自动排序。

3.1.6豆瓣数据抓取

豆瓣数据的抓取同其他平台的数据抓取大体类似，先依据豆瓣开放平台进行影视类数据获取依赖搜索接口进行内容查找定位，返回 id 后进行豆瓣内容获取。

各频道需要的豆瓣字段如表 3-1 所示：在前文所述 xml 中，所有字段都是必

填项，且都有内容；抓取回来的的豆瓣数据需要和单部影片进行一一对应，这些豆瓣字段优先原有字段进行展示（即有豆瓣 用豆瓣）。

电影	电视剧	动漫	综艺
大海报	大海报	大海报	大海报
主演	主演	主演	主演
导演	导演	导演	导演
语言	语言	剧情简介	剧情简介
豆瓣 tag	豆瓣 tag	豆瓣评分	豆瓣评分
剧情简介	剧情简介	豆瓣 tag	豆瓣 tag
豆瓣评分	豆瓣评分		
剧照列表↓	剧照列表↓		
【图片 id	【图片 id		
图片展示页 url	图片展示页 url		
图片地址，icon 尺寸	图片地址，icon 尺寸		
图片地址，image 尺寸	图片地址，image 尺寸		
图片地址，thumb 尺寸	图片地址，thumb 尺寸		
图片地址，cover 尺寸】	图片地址，cover 尺寸】		
影评列表↓	影评列表↓		
【影评 id	【影评 id		
影评名	影评名		
影评 url	影评 url		
发布日期	发布日期		
更新日期	更新日期		
条目 id	条目 id		
上传用户	上传用户		
摘要	摘要		
影评评分	影评评分		
有用数	有用数		
无用数	无用数		
评论数】	评论数】		

表 3-1 各频道豆瓣所需字段

上表标底色的内容为定时更新字段，不可修改；剩余字段均可修改。如果有豆瓣评分默认采用豆瓣评分（豆瓣评分属于不可修改字段）；管理员可以选择采用自有评分，自有评分属于可修改字段。

3.1.7豆瓣数据匹配

豆瓣的数据需要与我们自有数据形成一一对应的关系。依赖豆瓣搜索接口进行内容查找定位，归一化规则：用我们自有标题搜索后，豆瓣如果只有一部电影结果，直接相互对应上。如果有多个结果：

（1）选取标题相同+和我们自有年份相同的直接匹配

（2）选取标题相同+和我们自有导演相同的直接匹配（如果导演有多人时，其中一人完全匹配即视为完全匹配）

（3）选取标题相同+和我们自有主演 2 个以上相同的直接匹配

对于已经和豆瓣匹配上的数据，不再变化和不再重新匹配，只需要定时更新评分、影评和剧照。对于没有和豆瓣匹配上的数据，下次更新时再次进行匹配，直到它匹配上为止（动漫不需要进行）。

对于豆瓣数据人工干预，在后台电影、电视剧、动漫、综艺列表，如果没有对应豆瓣数据的影片，可以新增豆瓣 id 进行强制匹配，并当时进行一次完整的数据抓取。下次抓取时，该影片不再进行与豆瓣的按规则匹配，而是强制指定豆瓣 id 对应到这部影片。

对于已经匹配上，但匹配错误的数据，可以人工删除豆瓣 id 信息后，重新填入上述 id 进行强制匹配。

3.1.8数据排序

各个视频的数据排序根据视频类型所制定的策略如下：

（1）电影数据排序：热门排序（人工排序＞百度策略＞热门策略）、最新（人工排序＞入库时间）、好评（人工排序＞评分策略）

而其中的百度策略为：http://top.baidu.com/rss_xml.php?p=movie 标题与自有电影标题进行匹配，如果自有标题有多个重复时，取年份最新的。百度排行榜累加算法（此处以代码为准），今天 top50 和昨天的 top50，其中 20 部重复出现的，他们在第二天得分乘以 0.9，这样得出一个 top80 排序，依此类推。热门策略为（此处以代码为准）：多部电影排序，用其默认播放的站内相对排名得分（占 80%）+更新时间得分（占 20%）算出其 hot 得分，hot 得分越高的越靠前。评分策略为：有豆瓣评分大于没有豆瓣评分的；前面豆瓣评分的按照豆瓣评分排序，后面自有评分按照 10 分和 9.9 分、0 分的排在最后、从 9.8 分的开始排序。

（2）电视剧数据排序：热门排序（人工排序＞百度策略＞热门策略）、最

新（人工排序＞入库时间）

其中的百度策略为：http://top.baidu.com/rss_xml.php?p=tv 标题与自有电视剧标题进行匹配，如果自有标题有多个重复时，取年份最新的。每日替换，不需要进行累加算法。热门策略为（此处以代码为准）：多部电视剧排序，用其默认播放的站内相对排名得分（占 80%）+更新时间得分（占 20%）算出其 hot 得分，hot 得分越高的越靠前。

（3）动漫数据排序：热门排序（人工排序＞百度策略＞热门策略）、最新（人工排序＞入库时间）

其中的百度策略为：http://top.baidu.com/rss_xml.php?p=katong 标题与自有动漫标题进行匹配，如果自有标题有多个重复时，取年份最新的。每日替换，不需要进行累加算法。热门策略为（此处以代码为准）：多部动漫排序，用其默认播放的站内相对排名得分（占 80%）+更新时间得分（占 20%）算出其 hot 得分，hot 得分越高的越靠前。

（4）综艺数据排序：热门排序（人工排序＞百度策略＞热门策略）、最新（人工排序＞入库时间）

其中的百度策略：http://top.baidu.com/rss_xml.php?p=dianshi 标题与自有综艺标题进行匹配，如果自有标题有多个重复时，取年份最新的。每日替换，不需要进行累加算法。热门策略为（此处以代码为准）：多部综艺排序，用其默认播放的站内相对排名得分（占 80%）+更新时间得分（占 20%）算出其 hot 得分，hot 得分越高的越靠前。

（5）动漫数据排序：热门排序（人工排序＞百度策略＞热门策略）、最新（人工排序＞入库时间）

百度策略：http://top.baidu.com/rss_xml.php?p=katong 标题与自有动漫标题进行匹配，如果自有标题有多个重复时，取年份最新的。每日替换，不需要进行累加算法。

热门策略（此处以代码为准）：多部动漫排序，用其默认播放的站内相对排名得分（占 80%）+更新时间得分（占 20%）算出其 hot 得分，hot 得分越高的越靠前。

入库时间：入库越新的排在最新的前面，多家站点归并后的以默认播放的入库时间为准。

（6）综艺数据排序：热门排序（人工排序＞百度策略＞热门策略）、最新（人工排序＞入库时间）

百度策略：http://top.baidu.com/rss_xml.php?p=dianshi 标题与自有综

艺标题进行匹配，如果自有标题有多个重复时，取年份最新的。每日替换，不需要进行累加算法。

热门策略（此处以代码为准）：多部综艺排序，用其默认播放的站内相对排名得分（占 80%）+更新时间得分（占 20%）算出其 hot 得分，hot 得分越高的越靠前。

入库时间：入库越新的排在最新的前面，多家站点归并后的以默认播放的入库时间为准。

3.1.9数据分类

数据的分类需要可修改，且修改后在下次运算时生效。后台展示映射和修改后的分类。其实就是用正则表达式，根据映射规则，把获取到的数据和我们的分类一一对应，这其实也是数据的整合，先对数据进行检索，然后再将其他视频网站的分类替换成我们的分类，在方便用户查找同时，丰富我们网站自身的独特性，提高对同类型网站的竞争力。

（1）电影的分类主要是根据类型（爱情,喜剧,动作,恐怖,悬疑,科幻,灾难,战争,剧情,动画,音乐,纪录片,伦理,惊悚,奇幻,警匪,微电影,其他 ）和地区（华语,美国,欧洲,韩国,日本,泰国,法国,其他）进行分类，具体映射标准参见表 3-2。

（2）电视剧的分类也是根据类型（爱情,喜剧,动作,恐怖,悬疑,科幻,灾难,战争,剧情,动画,音乐,纪录片,伦理,惊悚,奇幻,警匪,微电影,其他 ）和地区（华语,美国,欧洲,韩国,日本,泰国,法国,其他）进行分类，标准参见表 3-3。

（3）动漫的分类则是根据类型（动作，亲子，原创，热血，冒险，古代，未来，竞技，体育，搞笑，言情，校园，都市，魔幻，科幻，励志，剧情，悬疑 ，宠物，LOLI，益智，童话，真人，神话，其他）和地区（日本，美国，英国，大陆，台湾，韩国，其他）来分类，具体分类映射标准参见表 3-4。

（4）综艺的分类是根据类型（真人秀，亲子秀，脱口秀）和地区（美国，韩国，内地，港台，其他）进行分类的。

电影、电视剧、动漫中的地区或者分类，大部分的数据都能够找到自己的映射，部分进行映射匹配失败的数据，即标为其他。单部影片可能存在部分映射成功但部分映射失败的情况，那么它的分类中其他或地区其他都展示在最后。例如电影 A 映射后类型为 其他 剧情 动作，则实际展现为剧情、动作、其他。

下面三张表显示的是电影、电视剧、动漫的映射规则，由于综艺分类较少且与前三种基本相同，在此不做过多赘述。

视频站分类	对应我们的分类		视频站地区	对应我们的
传记	纪录片		大陆	华语
短片	微电影		香港	华语
犯罪	警匪		台湾	华语
功夫	动作		德国	欧洲
纪录	纪录片		意大利	欧洲
纪实	纪录片		俄罗斯	欧洲
军旅	战争		中国内地	华语
枪战	动作		中国香港	华语
探索	纪录片		中国台湾	华语
刑侦	警匪		内地	华语
艺术	音乐		西班牙	欧洲
爱情片	爱情			
喜剧片	喜剧			
恐怖片	恐怖			
科幻片	科幻			
灾难片	灾难			
战争片	战争			
剧情片	剧情			
音乐片	音乐			
奇幻片	奇幻			
警匪片	警匪			
传记片	纪录片			
犯罪片	警匪			
歌舞片	音乐			
功夫片	动作			
纪录片	纪录片			
纪实片	纪录片			
军旅片	战争			
魔幻片	奇幻			
枪战片	动作			

表 3-2　电影映射标准

视频站分类	对应我们的		视频站地区	对应我们的
爱情	言情		美剧	美国
犯罪	警匪		泰剧	泰国
搞笑	喜剧		大陆	内地
军事	军旅		中国大陆	内地
情感	言情		中国香港	香港
时尚	都市		中国台湾	台湾
时装	都市		港剧	香港
刑侦	警匪		中国内地	内地
罪案	警匪		台剧	台湾
青春剧	青春		韩剧	韩国
家庭剧	家庭		TVB	香港
军旅剧	军旅		日剧	日本
言情剧	言情			
古装剧	古装			
武侠剧	武侠			
偶像剧	偶像			
宫廷剧	宫廷			
历史剧	历史			
年代剧	年代			
都市剧	都市			
伦理剧	伦理			
动作剧	动作			
神话剧	神话			
战争剧	战争			
搞笑剧	喜剧			
情感剧	言情			
时尚剧	都市			
时装剧	都市			
罪案剧	警匪			

表 3-3 电视剧映射标准

视频站地区	对应我们的
内地	大陆
中国大陆	大陆
中国	大陆
中国台湾	台湾
中国内地	大陆

表 3-4　动漫映射标准

这部分正则表达式的应用其实就是正则表达式的替换，找到分类“ABC”，然后把内容替换为我们所需的“EDF”，只是由于数据量大，根据映射标准表将分类一一替换。

3.2　系统的非功能性需求

结合该系统的功能特点，其非功能行需求如表 3-5 所示。

非功能性需求	详细要求
安全性	大陆视频信息统一调用接口获取，传输过程进行加密处理，保证获取的数据信息的安全性
稳定性	系统里的数据实时备份，不存在一台或者几台服务器出现故障，导致数据丢失或系统无法工作的情况出现。
可靠性	每季度可容许的服务意外中断次数为 1 次； 每次课容许的最大服务意外中断时间为 30min.
可用性	平均每季度用户可访问的时间不低于该季度的 99.99%
易用性	页面布局合理、清晰，用户能快速简便的找到自己所需的视频
可扩展性	系统支持接入新的业务，基本的代码框架满足新功能的开发，原功能不受影响。

表 3-5　系统非功能性需求

3.3　本章小结

本章从系统的功能性需求和非功能性需求这两个方面对系统进行了详细的分析，介绍了各个部分对于影视数据的处理要求，同时也对非功能性需求进行了

分析，包括安全性、稳定性、可靠性、可用性、易用性、可扩展性，保证了系统的质量。

4 系统概要设计

本章将要介绍导航网站数据处理的架构设计，主要包括系统的架构设计和各类型视频数据的处理策略。

4.1 整体架构设计

本导航网站采用 MVC 的框架设计模式，把传统的输入、处理、输出功能映射在一个逻辑的图形化用户界面的结构里，其框架参见图 4-1。

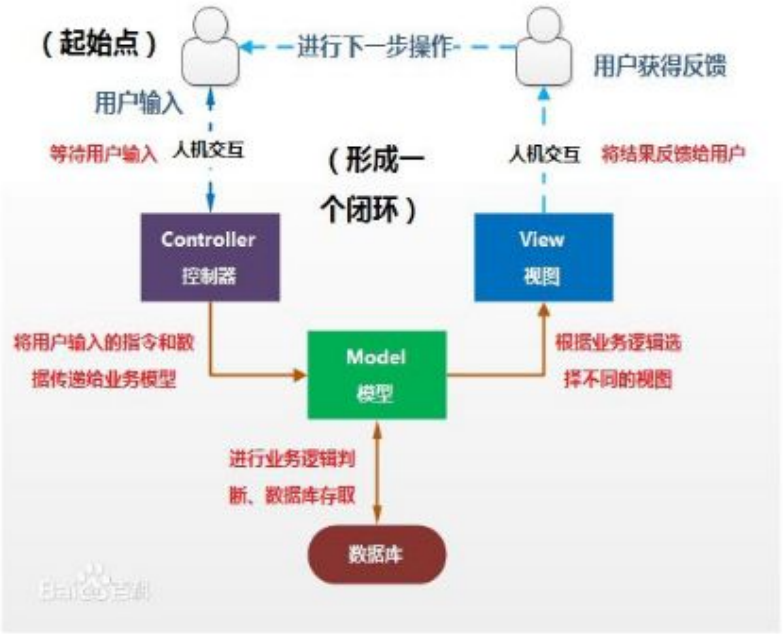

图 4-1 MVC 模式

本网站在数据处理上包括数据抓取、更新、修改、删除、归并、排序、分类等几个部分。先通过 XML 获取其他视频网站的数据，在解析数据之后判断数据是否存在于删除接口中，如果是，则将其加入 delete 表，否则，要通过归并将数据进行整合，再入库，入库之后根据不断获取的数据即时更新、修改、删除。数据库里的数据在输出到网站页面之前通过定义好的规则进行排序和分类，然后显示到各个页面上，见图 4-2。

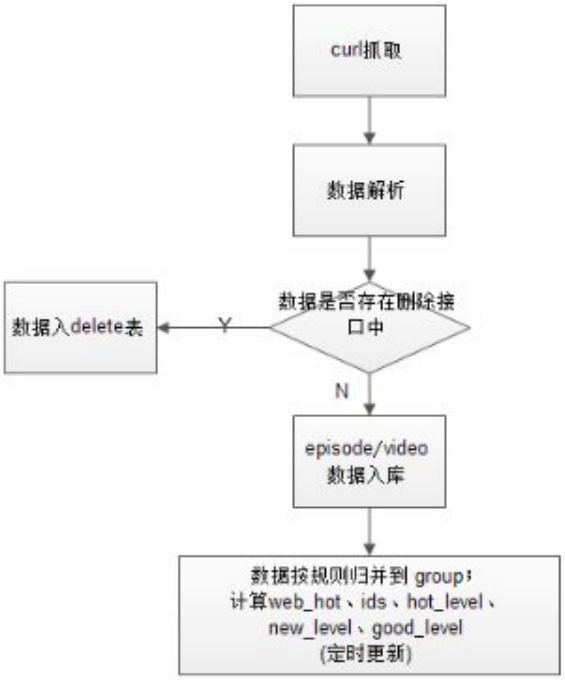

图 4-2 数据处理流程图

数据抓取是从大型数据库中挖掘有效信息，其基本过程和步骤参见图 4-3：

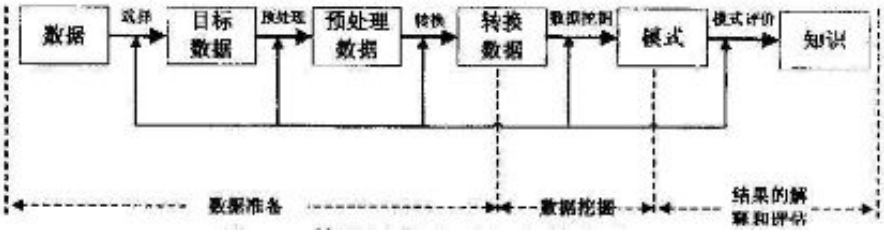

图 4-3 数据抓取的过程和基本步骤

在数据挖掘的过程中不需要使用数据库管理系统，建立一个独立的数据集即可，由于网站主要考虑用户体验度，因此要获得描述性的、容理解的数据，所以我们选择采用规则表示的挖掘方法，而不是精神网络的方法。

系统在接收到抓取来的 XML 数据之后，会对原始的数据进行包装，并对初始数据里的一些不符合要求的数据或特殊字符进行替换，从而避免无法处理特殊字符或者过多无用数据占用数据库导致崩溃问题的发生。

进行数据归并时，首先需要从配置文件中读取预先设定好的归并策略，然后根据制定的规则对接收到的数据进行匹配，成功归并数据后，按照归并规则将处理好的数据发送给数据处理模块中的不同实例入库。

在数据处理过程中，数据搜索的过程入图 4-4 所示。

图 4-4 coreseek 数据搜索流程

4.2 数据处理策略的设计

对于所有的视频而言，listlink、playlink 重复整部剧集不能入库，必填字段为空不能入库，字段不按照规定格式。首先，各字段必须在接口模板要求的字节数之内，超出的不入库。所以在设计正则表达式语句的时候根据文字符号和字符集等制定，文字符号写作<<a>>（注：正则表达式引擎对大小写十分敏感，因此需要将引擎设定好忽略大小写），字符集则是由“[]”括起来的字符集合，选择匹配一个或者多个字符。

正则表达式可看作由不同类型的片连接而成的，分片结果参见表 4-1.其中的片是闭包、计数结构、集合和它修饰的结构，简化匹配算法、提高性能。

	Regular expression
1	【.*】contenttype=【[^\r\n\x3b\x38]{100}】
2	【.*】\x28\s*name\s*\x22【[^\x22]{260,}】
3	^TEST\s+【[^\n]{100,}】
4	【.*】cache_lastpostdate\[【[^\]]+】\]=【[^\x00\x3B\x3D]{30}】
5	【.*】\{\s*【(\x27[^\x27]{1024,}|\x22[^\x22]{1024,})】
6	【.*】\x2Fcgi\x2Flogurl\.cgi【.*】User-Agent\x3A【[^\r\n]*】 MyPost【.*】form-data\x3B\x20name=\x22pid\x22【.*】internal

表 4-1 正则表达式的分片结果

由此设计了根据文字符号和字符集处理的各个视频类型的处理策略。

4.2.1电视剧数据处理策略设计

电视剧主要是针对导演、标题、语言、主演、时间等信息进行入库和不入库的判断，设计规则如下：

（1）<director>、<region>、<type>、<language>、<definition>、<otherName>不用半角;分隔的，不入库。

（2）<showTime>不是 yyyy 格式四位数字的，不入库。

（3）<hot>、<seasonId>、<length>不是≥1 的整数，不入库。

（4）<score> 不是浮点数，不入库。

（5）<updateTime>、<resourceTime>不是 YYYY-MM-DD hh:ii:ss 格式的不入库。

（6）<seq>、<nowEpisode>不是≥1 的整数，不入库；<totalnumber>只有当≥1 或=-1 的情况下可以入库。且当<nowEpisode>≠最大<seq>时不能入库。<nowEpisode>=最大<seq>，且<totalnumber>≠-1 的条件下：<status>=1 时，<nowEpisode>≠<totalnumber>时，不入库。<status>=2 时，<nowEpisode>≥<totalnumber>时，不能入库。

4.2.2电影数据处理策略设计

电影数据处理大体与电视剧相同，其设计规则如下：

（1）<director>、<region>、<type>、<language>、<definition>、<otherName>不用半角;分隔的，不入库。

（2）<showTime>不是 yyyy 格式四位数字的，不入库。

（3）<length><![CDATA[6167]]></length> 不是≥1 的整数，不入库。

（4）<score> 不是浮点数，不入库。

（5）<resourceTime、<updateTime>不是 YYYY-MM-DD hh:ii:ss 格式的不入库。

（6）< hot>、<seasonId>、<length>不是≥1 的整数，不入库。

4.2.3综艺数据处理策略设计

综艺数据处理的规则类似，但是在<detail>和<date>方面略有不同，其规则如下：

（1）<region>、<type>、<language>、<definition>、<otherName>、<host>、<starring>不用半角;分隔的，不入库。

（2）<showTime>不是 yyyy 格式四位数字的，不入库。

（3）<score>不是浮点数，不入库。

（4）<updateTime>、<resourceTime>不是 YYYY-MM-DD hh:ii:ss 格式的不入库。

（6）<hot>、<length>不是≥1 的整数，不入库。

（7）<date>不是八位数字的，不入库。

（8）综艺<detail>内有重复的<date>，不能入库， 即每一部综艺不能有期数重复的两集。

4.2.4动漫数据处理策略设计

由于很多动漫存在即时更新的情况，所以动漫数据处理规则如下：

（1）<role>、<starring>、<director>、<region>、<type>、<language>、<definition>、<otherName>不用半角;分隔的，不入库。

（2）<showTime>不是 yyyy 格式四位数字的，不入库。

（3）<score>不是浮点数，不入库。

（4）<seasonId>、<hot>、<length> 不是≥1 的整数，不入库。

（5）<seq>、<nowEpisode>不是≥1 的整数，不入库；<totalnumber>只有当≥1 或=-1 的情况下可以入库，且当<nowEpisode>≠最大<seq>时不能入库<nowEpisode>=最大<seq>，且<totalnumber>≠-1 的条件下：<status>=1 时，<nowEpisode>≠<totalnumber>时， 不入库<status>=2 时，<nowEpisode>≥<totalnumber>时，不入库。

（6）<updateTime>、<resourceTime> 不是 YYYY-MM-DD hh:ii:ss 格式的不入库。

4.3 本章小结

本章主要从系统的架构设计和各类视频数据的处理策略设计两个方面介绍了系统的概要设计，描述了数据处理的主要流程和方式，介绍了基于什么设计的处理策略以及最终制定的处理策略。

5 系统详细设计

本章主要介绍系统的详细设计，通过介绍模块的功能及功能的实现方法等角度对数据处理的各个模块进行描述。

5.1 功能模块详细设计

本节主要是对数据处理部分各个功能模块的详细设计进行介绍，主要包含数据的抓取、更新、删除、归并等功能。

5.1.1数据抓取模块

将各 xml 中的指定站点 webName 和 webSiteUrl，和我们登记的站点进行对应，不在登记站点列表内的 webname 或 websiteurl 则 xml 无法入库。webName 和 webSiteUrl 确定后，抓取各站点来源的数据。

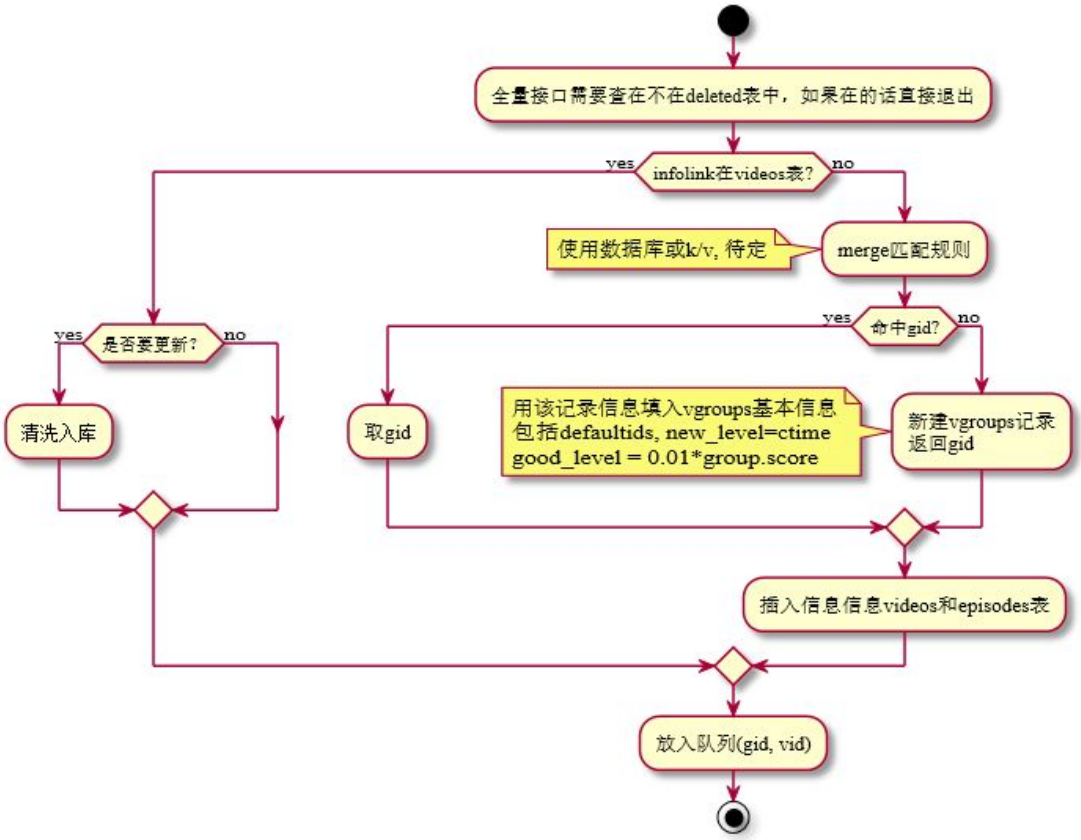

图 5-1 数据抓取详细流程

数据抓取方式如图 5-1，如果数据在 delete 表中，那么直接退出，否则判断 infolink 是不是在 videosbi 表中，若在，那么判断是否需要更新，如果需要

更新就清洗入库，如果不需要就放入队列（gid,vid）。如果不在 videos 表中，那么使用数据库判断是否遵从 merge 匹配规则，如果命中 gid，就取 gid 插入信息 videos 和 episodes 表,放入队列 gid,vid ,如果不命中 gid,则新建 vgroups 记录返回 gid。

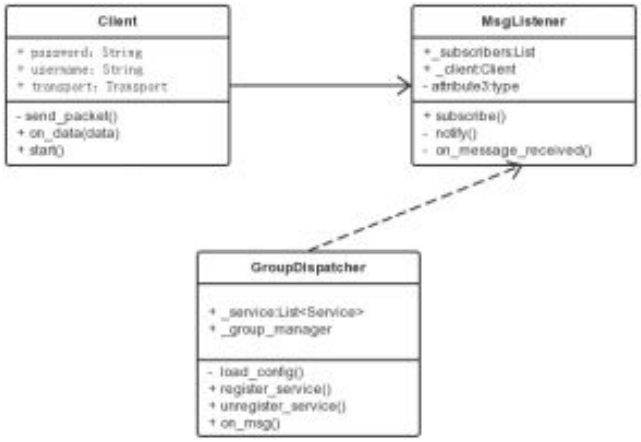

图 5-2 数据抓取模块类图

影视网站文本数据的挖掘是在计算语言学、统计数理分析的理论基础上，把信息检索技术和机器学习相结合，提取有效信息，处理过程如图 5-3 所示。

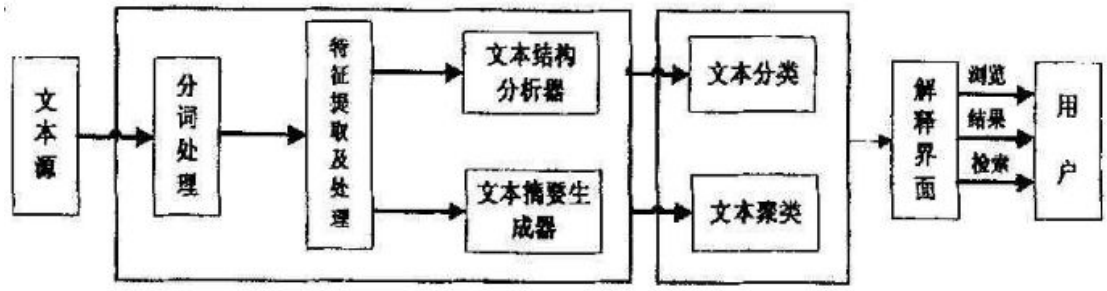

图 5-3 影视网站文本挖掘的处理过程

文本挖掘的过程中需要分词、特征提取、分析超链接结构，抽取文本的词条，用公式：

$$W_{ik}=\frac{tf_{ik}*\log\left|N/n_k+0.5\right|}{\sqrt{\sum_{k=1}^{n}\left|(tf_{ik})^2*\log^2\left|N/n_k+0.5\right|\right|}} \qquad (5\text{-}1)$$

式计算词条权重，对文本中的标题、日期、简介等特征提取，考虑到 HTML 文档的特点，对标题等文本赋予较高权值，来提高提取的效率。

在抓取的同时，在把数据显示到前端前，还必须先根据筛选数据逻辑图 5-4 处理数据。

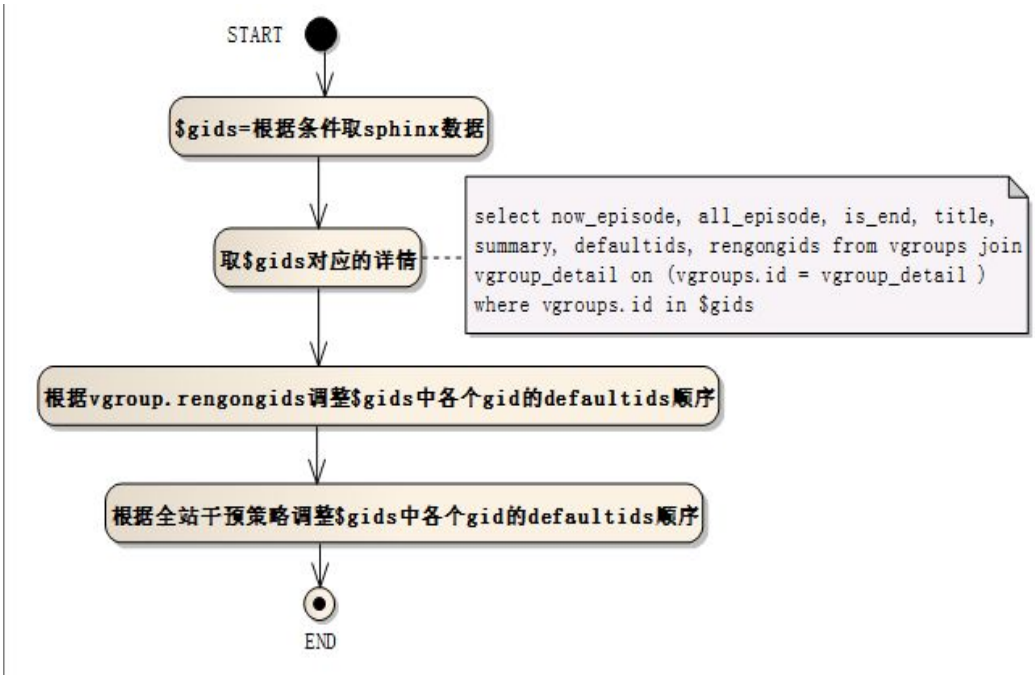

图 5-4 筛选数据逻辑图

前端的首页，轮播图部分，左上角为 5 个轮播大图，显示标题及图片。可通过下方轮播及左右的箭头进行切换，本身图片可 5 秒钟自动轮播一个。数据来源：碎片编辑。其他区域共 8 个小图，展示标题及图片，均可点击，且有一定的 hover 效果。数据来源：碎片编辑。其数据来源于管理系统发布的 reids，其中 Key 在 focus_recommend_vtype，轮播大图的信息是取数据直接展现。

图 5-5 首页轮播大图

筛选栏的类型、地区都是固定不变的，明星的数据来源于管理系统发布的 reids，key 在 star_recommend_vtype，年份为当前年份向后，一共 8 年。热门专题内容与线上一致，主要有总榜、电影、电视剧、综艺、动漫。数据来源：总榜是碎片，其他频道为频道列表页的热门前十的数据，这个模块的数据来源于管理系统发布的 reids，key 在 zt_recommend_vtype。除热门专题外，所有点击都去到筛选页面，链接拼接规则参加（三.筛选页）。左侧推荐则是包括大家都在看，

内地，港台，欧美等。数据来源于管理系统发布的 reids，key 在 fillhole_vtype_rtype，json 数据，按顺序展现即可，不需要作其他的处理。明星推荐则是来源于管理系统发布的 reids，key 在 tv_star_rank 不同地区以 typeid 区分。Json 是 hash 数组，key 是 typeid，val 是该地区明星数据。热播剧榜单包括内地、港台、泰国等，数据来源于定期生成的榜单 reids，key 在 tv_hotrank_areatype,榜单是按热门分数进行排序的，和大首页使用的榜单一致。

图 5-6 筛选栏界面

电视剧部分推荐当前热门的电视剧，右侧上方展示电视剧的热门分类，为“热门、最新、内地、韩剧、港剧、台剧、泰剧、美剧、更多”。默认为热门 tab 下，用户点击其他 tab 才会触发其他 tab 的内容展示。

筛选区调用的分类数据与电视剧列表页一致，用户点击某个分类则新窗口打开跳转到该筛选条件下的电视剧列表页，默认为全部条件。右侧展示热门专题，数据来源：碎片编辑。普通版时热门专题区域不显示。

每个 tab 下方共展示 12 个电视剧。每行 4 个，共 3 行。电视剧采用横图海报，海报来源为后台新增横图字段。编辑人工编辑后在前台展现。海报底部有半透明浮层展示更新情况，两种情况“更新至 XX 集、XX 集全”。海报下方展示电视剧的标题，标题及海报需要有一定的 hover 效果。数据来源：电视剧列表页的前 12 个电视剧。其他 tab 的数据来源为一定筛选条件下的前 12 个电视剧。

右侧展示电视剧的 4 个分类筛选，均为文字展现，数据来源：电视剧列表页的前四个分类筛选。点击更多进入电视剧列表页，点击各个分类进入该筛选条件下的电视剧列表页。

图 5-7 电视剧推荐栏

电影部分推荐当前热门的电影，右侧上方展示电影的热门分类，为“热门、最新、喜剧、动作、恐怖、爱情、欧美、战争、伦理、黄渤、更多”。默认为热门 tab 下，用户点击其他 tab 才会触发其他 tab 的内容展示。

每个 tab 下方共展示 8 个电影。每行 4 个，共 2 行。第一行采用竖版海报，第二行采用横版海报。竖版海报的来源为豆瓣数据的大图，编辑可在后台修改，横版海报的来源为后台新增的横图 2 的字段，编辑可在后台添加修改。海报底部有半透明浮层展示播放画质和电影时长。海报下方展示两行文字，分别为电影标题及电影的一句话评论。海报和标题需要有 hover 效果，点击进入该电影的中间页。一句话评论不可点。

数据来源：热门和最新的数据来源为电影频道的“热门推荐”和“网络首播”，其他 tab 分类下的数据来源为最热电影列表页该分类的前 8 个电影。

右侧展示电影的 4 个分类筛选，均为文字展现，数据来源：电影列表页的前四个分类筛选。点击更多进入电影列表页，点击各个分类进入该筛选条件下的电影列表页。如果该区域长度较短，可进行碎片编辑。

图 5-8 电影推荐栏

综艺的标题右侧展示几个 tab，为“热门、大陆、港台、日韩、更多”，默认为热门，每个 tab 下方有 8 个横版海报，图片来源为后台的横图 2 字段。海报底部展示半透明浮层，上方展示更新期数，“2013-08-07 期”，下方两行文字，第一行为标题，第二行为描述，海报及标题有 hover 效果，点击进入综艺中间页，描述为静态文字。

数据来源：热门 tab 为综艺列表页的前 8 个。其他 tab 为该分类下的前 8 个。点击更多进入综艺列表页。

在下方展示直播节目，展示卫视及图表，每个卫视下有两个热门综艺节目，该部分为调用综艺首页下方的卫视强档的前 8 个，卫视强档的栏目为碎片编辑。

右侧展示热门综艺，文字链，碎片编辑。娱乐看点，文字链，碎片编辑。多余区域为专题图位置。

右侧里的卫视独播数据来源于管理系统发布的 reids，key 在 tv_weishi_hotplay，

图 5-10 前端电视剧集数显示图

图 5-10 中电视剧的的集数 index 信息取自 now_episode,链接统一写 episodejump.php?vid=$vid&index=$index&vtype=$vtype&maxvid=$maxvid&gid=gidEpisodejump.php 跳转到具体播放链接的逻辑如图 5-11 所示：

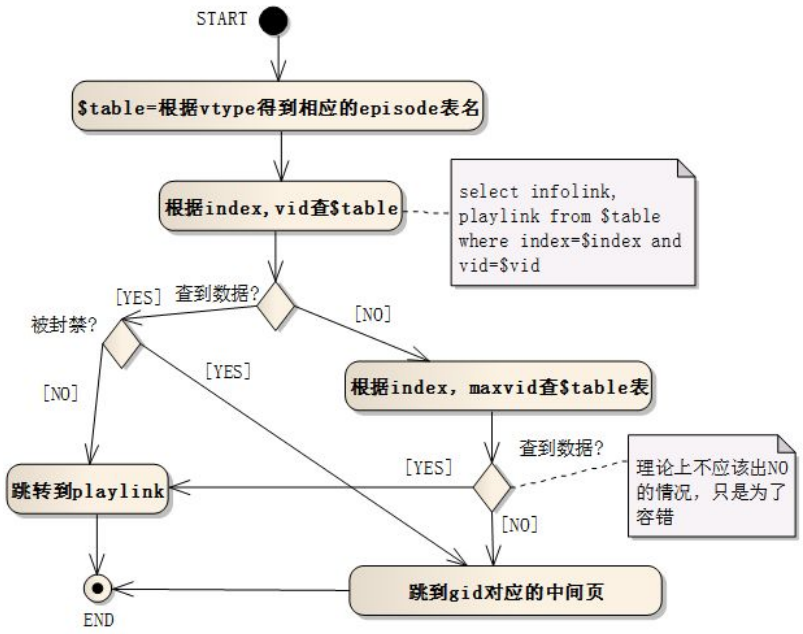

图 5-11 链接跳转逻辑图

对于追剧部分来说，本地数据记在 cookie 中，取数据实现方式：异步取数据，其接口名称为 tvFollow.php，Post 参数是 val:cookie 中的 json，响应到 html 页面片。接口需要做简单的 refer 验证，还要考虑增加本地缓存。（注：

tvFollow.php 取数据逻辑：episode_dianshiju。Select index, playlink from episode_dianshiju where vid = $vid order by index limit 20; 在前端展现时，如果最大集数 allseq 和最新一集相等，则显示已完结，否则显示未完结。如果最后一条记录的 index > 1，则显示更多。

图 5-12 追剧界面

其中的追剧策略设计为：当用户点击具体集数时，记录：当前观看的集数，gid 对应的 title，vid，总集数。数据记录在 cookie 中 Key: _tv_follow，val:[{seq，title，vid，allseq}，...]当 val 数组大于 6 项时，删除最后一项。

Vid 上取 isend，episode 表，根据 vid 取播放连接，最多提取 20 条。对于已经点过的剧集靠链接的 visited 记录，显示不同的底色。在追剧过程中点击顶踩，根据图 5-13 获取数据，取得顶踩数据并显示在前端则如图 5-14 所示。

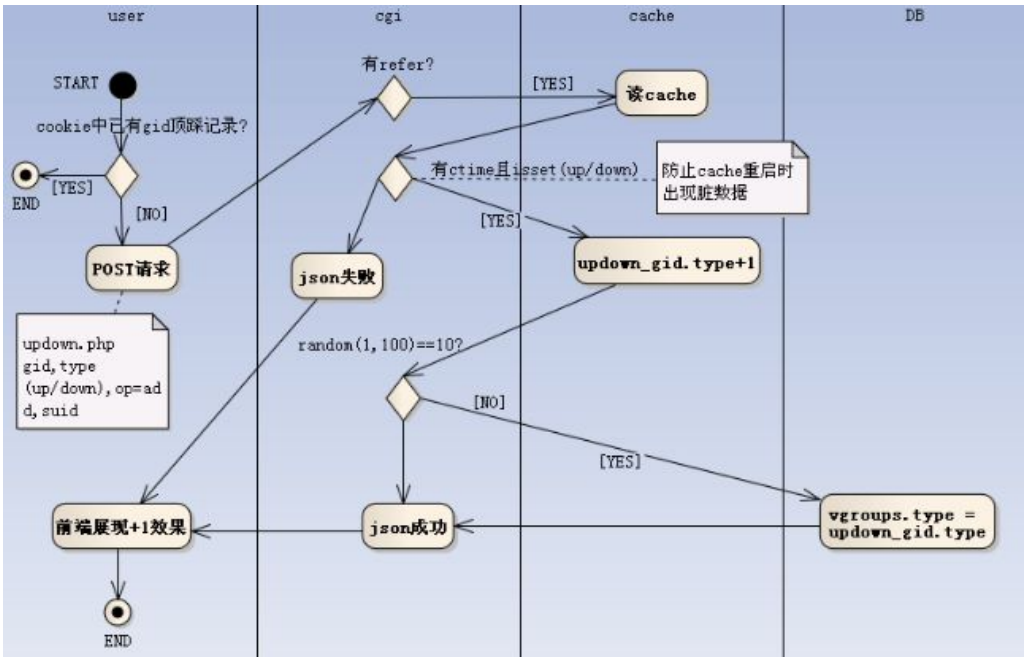

图 5-13 点击顶踩过程图

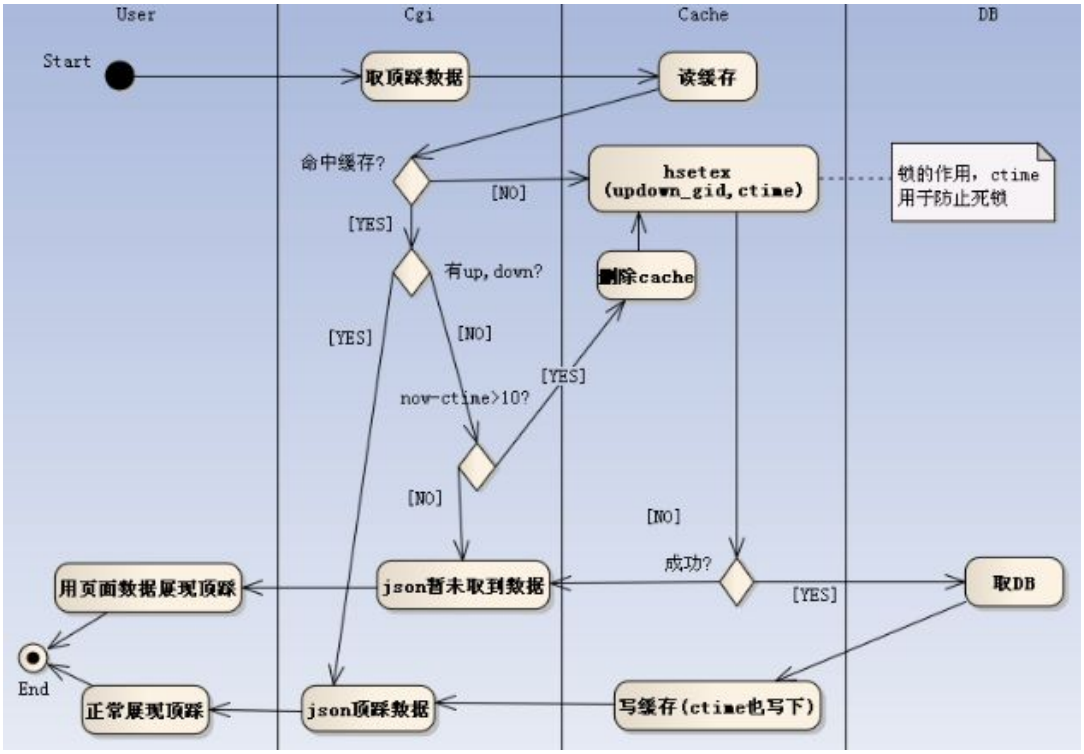

图 5-14 取顶踩数据过程图

这些顶踩数据会有异步显示（因为有缓存，同步会有问题）。接口在updown.php。Post 为 gid, op=get, suid。Json 方式返回缓存中的相应 gid 的实际顶踩数据{gid:{up,down}}

5.1.2数据更新与修改模块

来自各个网站的视频数据每天都需要定时进行更新，以保证网站视频信息的完整、即时。更新过程中，为了简化算法，将正则表达式中 ET2 类型的片分割部分，再组合，它分片和组合的过程如图 5-15.

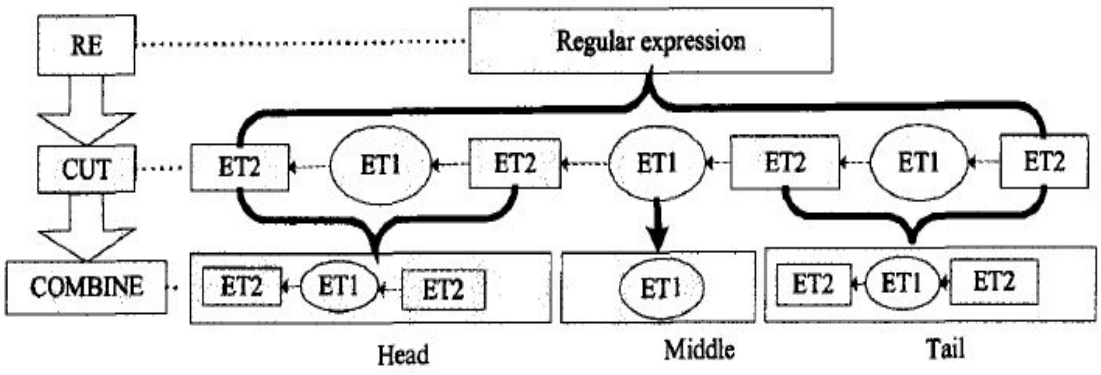

图 5-15 正则表达式分片及组合过程图

利用 RECCADR 算法，依次试探 ET2 在中部时各个部分的 DR 变化情况，如果头尾部 DR 大幅降低，说明找到使得正则表达式歧义匹配的部分（注：DR 代表正则表达式膨胀率，不同的正则表达式膨胀率和复杂程度都不同）。它按照 DR 的高低顺序选择正则表达式加入 DFA.这个算法能够实现把任意正则表达式规则集合转换为 DFA 的功能，在 DFA 远小于正则表达式个数的时候，它的时间复杂度为零，加快了数据处理的速度。

Regular expression	Before		After					
	DFA	DR	Head		Middle		Tail	
			DFA	DR	DFA	DR	DFA	DR
.*\sRENAME\s•[^\n]*•\s\{	49	0.887	24	0.2	6	-0.455	11	-0.214
.*\sLOGIN\s•[^\n]{100}•	>10 000	>10 000	22	0.158	104	-0.046	N/A	
•.*\s{23}•DIMBUS\s+Server\s+v\d+\x2E\d+	938	15.456	N/A		27	-0.156	37	0.121

图 5-16 正则表达式分片 DR 和 DFA 的关系

由于需要更新的部分较多，这里以豆瓣信息更新为例，先要查询 doubanid 且年份是近五年的 gid（select gid from douban where year<year_now-5），然后用每 doubanid 查询评论、评分、tags 等信息，再更新豆瓣表。具体如图 5-17。

图 5-17 豆瓣更新流程图

针对上述正则表达式的特点设计了字段修改、覆盖的要求，先检索出符合要求的文本，然后集合最新抓取的数据，对原有数据进行替换，更新 douban 表。

字段的更新和覆盖规则遵从表 5-1.表 5-2，表 5-3 和表 5-4.

电影字段	能否修改	能否被覆盖
webName	×	
webSiteUrl	×	
listLink	×	
workName	√	×
director	√	×
name	√	×
role	√	×
region	√	×
showTime	√	×
length	√	×
type	√	×
imageLink	√	×
imageLink2	已去掉	
imageLink3	√	×
introduction	√	×
comment	√	×
playLink	×	
mplayLink	×	
score	√	×
language	√	×
definition	√	×
seasonId	√	×
otherName	√	×
source_type	×	
swfUrl	×	
resourceTime	×	
status	√	×
updateTime	×	
hot	×	

表 5-1 电影字段更新和覆盖规则

电视剧字段	能否修改	能否被覆盖
webName	×	
webSiteUrl	×	
listLink	×	
workName	√	×
director	√	×
name	√	×
role	√	×
region	√	×
type	√	×
showTime	√	×
imageLink	√	×
imageLink2	√	×
hot	×	
score	√	×
language	√	×
introduction	√	×
seasonId	√	×
otherName	√	×
source_type	×	
status	√	√
nowEpisode	√	√
updateTime	×	
update	√	×
seq	×	
singleTitle	×	
singleLink	×	
mplayLink	×	
singleIntroduction	×	
swfUrl	×	
resourceTime	×	
singleThumbnails	×	
length	×	

表 5-2 电视剧字段更新和覆盖规则

综艺字段	能否修改	能否被覆盖
webName	×	
webSiteUrl	×	
listLink	×	
workName	√	×
region	√	×
showTime	√	×
type	√	×
score	√	×
language	√	×
tvStation	√	×
imageLink	√	×
imageLink2	√	×
introduction	√	×
definition	√	×
otherName	√	×
source_type	×	
host	√	×
updateTime	×	
update	√	×
hot	×	
date	×	
starring	√	×
singleTitle	√	×
singleLink	×	
mplayLink	×	
swfUrl	×	
resourceTime	×	
singleThumbnails	√	×
singleIntroduction	√	
length	√	×

表 5-3 综艺字段的更新和覆盖规则

动漫字段	能否修改	能否被覆盖
webName	×	
webSiteUrl	×	
listLink	×	
workName	√	×
imageLink	√	×
imageLink2	√	×
role	√	×
starring	√	×
director	√	×
region	√	×
type	√	×
showTime	√	×
score	√	×
language	√	×
introduction	√	×
definition	√	×
totalnumber	√	√
seasonId	√	×
otherName	√	×
source_type	×	
status	√	√
nowEpisode	√	√
updateTime	×	
update	√	×
hot	×	
seq	×	
singleTitle	×	
singleLink	×	
mplayLink	×	
swfUrl	×	
resourceTime	×	
singleIntroduction	×	
singleThumbnails	×	
length	×	

表 5-4 动漫字段的更新和覆盖规则

前端页面的排行榜和相关资讯的数据也是定时更新的，电视剧排行榜的数据来源：定期生成榜单-->redis，key：tv_hotrank_typetype，其中 type 为当前电视剧的第一个类型，按热度排序。相关资讯则是通过搜索接口异步，取 redis 数据。通过接口 zixun.php 获取参数 gid,vtype, title 以及 Key：zixun_vtype_gid，val，json 数据。如果 redis 没有，则穿透到搜索接口取数据。缓存 4 小时。即使搜索取回的是 0 条数据，也要有相应的 key，val 是空数

组。

在视频页面的猜你喜欢部分的数据获取更新策略是：有豆瓣 tag 时，按 tag 匹配个数排序，同个数以 hot 分排序。无豆瓣 tag 时，按分类，年份，地区匹配个数排序，同个数以 hot 分排序。首先通过 Sphinx 导入豆瓣 tag，分类，年份，地区词表。然后 SetRankingMode 使用 SPH_RANK_WORDCOUNT，SetMatchMode 再使用 SPH_MATCH_EXTEND2，其中 SetSortMode（SPH_SORT_ATTR_DESC, hot）如果有 douban，则@tag $t1|$t2|$t3，如果无 douban，则@(cate) $c1|$c2|$c2 @year $year @area $area。

推荐的同导演/主演部分的数据更新策略是：导演，主演匹配个数优先，个数相同，按热度分数排序。Sphinx 中导入导演，主演词表。SetRankingMode 使用 SPH_RANK_WORDCOUNT。SetMatchMode 使用 SPH_MATCH_EXTEND2。其中 SetSortMode（SPH_SORT_ATTR_DESC, hot）。

如果需要对豆瓣表的数据进行修改，那么取 doubanid 为空的 gid，用 title 做参数调用豆瓣的搜索接口，然后返回 subject_ids，迭代完成后判断 subject_ids 是否在 subject_ids 里，若否，则舍弃数据。若是，取 subject_id 里对应的信息，再判断豆瓣信息是否符合匹配规则，若否则返回，若是则用匹配到的 subject_id 获取豆瓣影评、剧照、tags 等信息，经过数据清洗之后插入豆瓣表。具体过程见图 5-17.

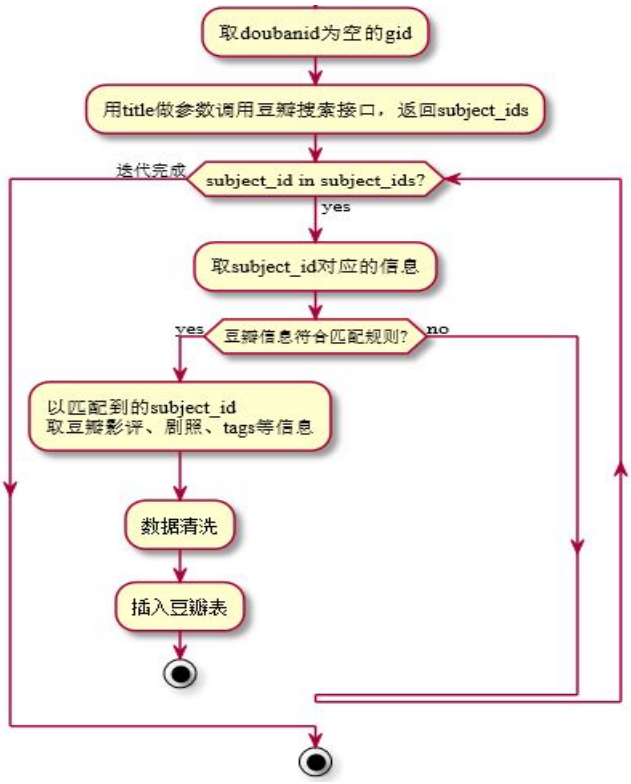

图 5-17 豆瓣数据修改流程图

如果要在后台手动抓取豆瓣信息，则需要先用添加的 doubaind 取详情、影评、剧照、tags 等信息，判断是否成功，如果成功，则在豆瓣表删除 gid=$gid 的记录，用新数据插入 douban 表，如果不成功，则没有后续步骤。

在中间页获取基本信息时，需按图 5-18 进行。

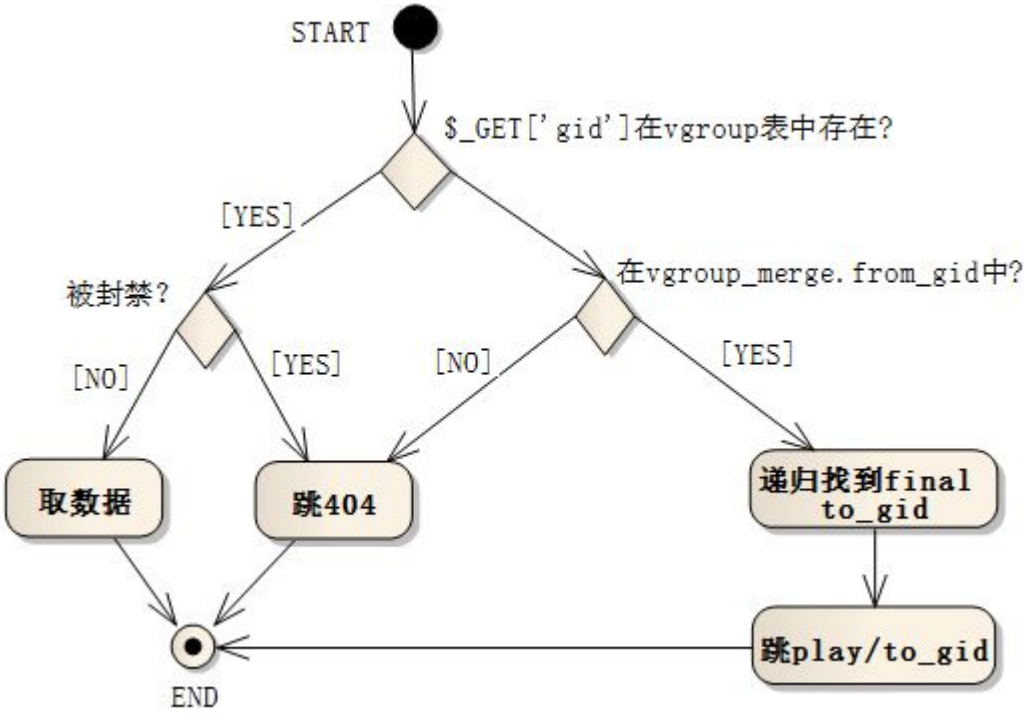

图 5-18 中间页获取基本信息过程图

基本信息的图片来自豆瓣大图，如果没有就从 vgroups.imagelink 中提取。

中间页的评分逻辑见图 5-19.

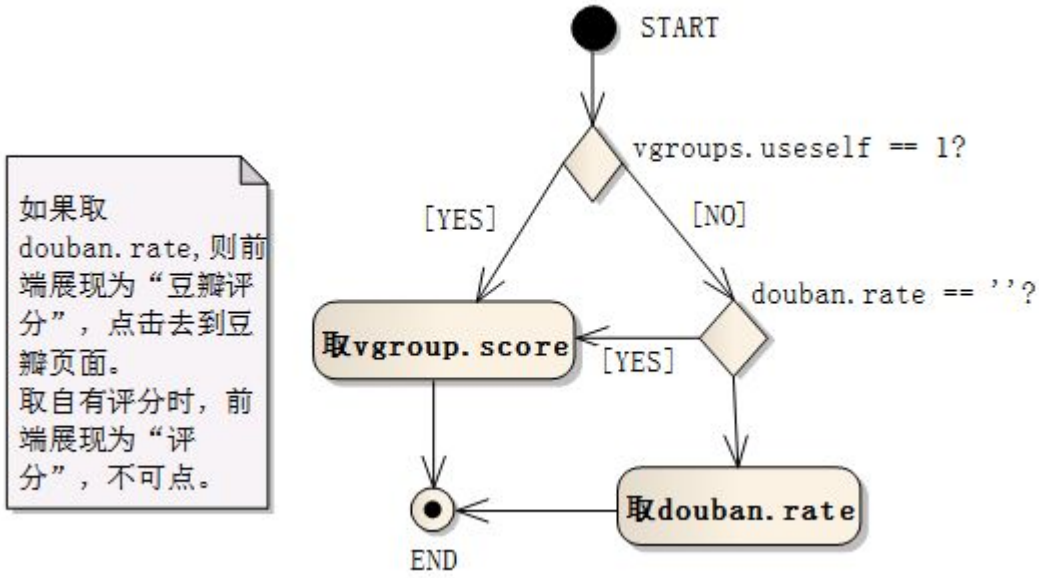

图 5-19 中间页评分逻辑图

取剧集和预告片的时候，当 vids 首位放源集数最多时，默认展现首位放源 tab，否则默认展现“全部”tab。如果剧集被封或删除，则相应剧集置灰。只在默认 tab 中展现预告片，同时切换 tab 时异步拉取相应播放源剧集数据及剧情信息。具体流程见图 5-20.

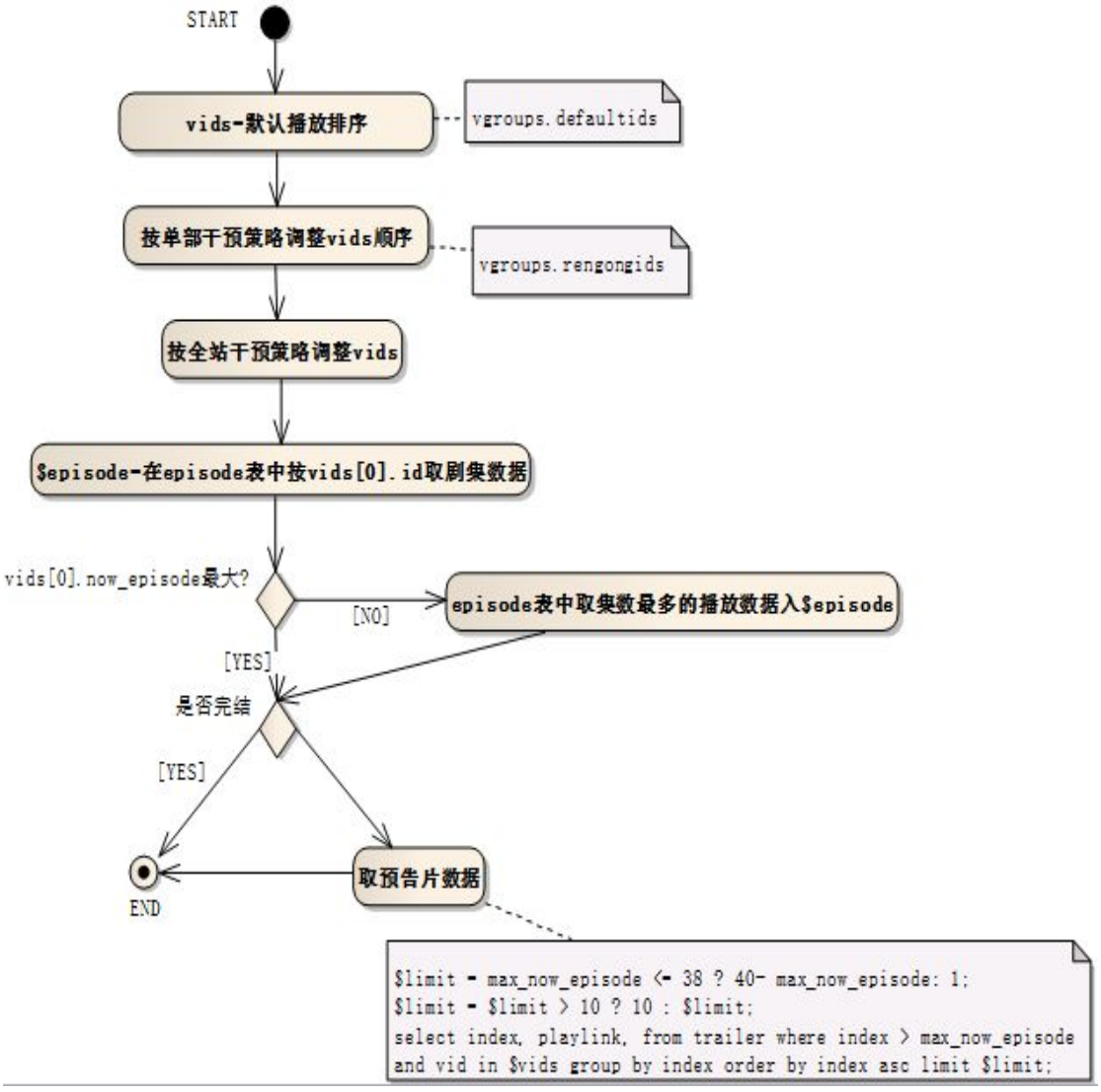

图 5-20 中间页取剧集流程图

相关资讯部分也需要定时更新，搜索接口异步，取 redis 数据，接口名为 zixun.php。GET 参数：gid,vtype, title。Key：zixun_vtype_gid，val，json 数据。如果 redis 没有，则穿透到搜索接口取数据。缓存 4 小时。即使搜索取回的是 0 条数据，也要有相应的 key，val 是空数组。这其中搜索结果的情况见表 5-5.

序号	情况	结果组成	说明
a)	命中多条合作数据	合作数据结果+视频搜索结果	
b)	命中一条合作数据，且该数据为电影或电视剧	合作数据结果+推荐短片+视频搜索结果	
c)	命中一条合作数据，且该数据为动漫或者综艺	合作数据结果+视频搜索结果	
d)	没有命中合作数据	视频搜索结果	
e)	合作数据和视频搜索均无结果	搜索无结果提示	
f)	只命中合作数据	只展示合作数据	

表 5-5 搜索结果情况图

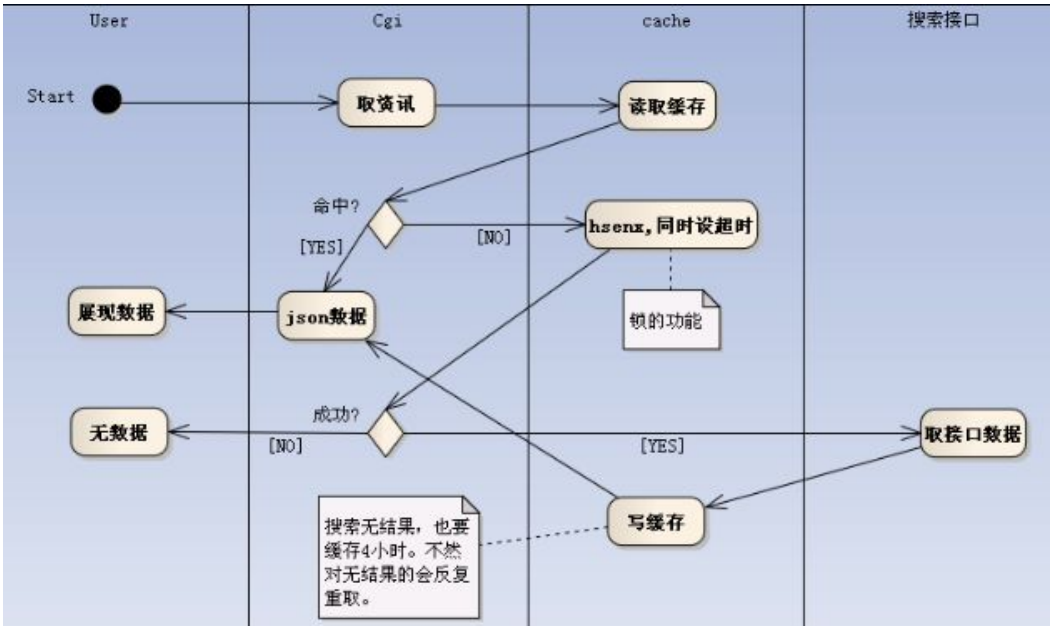

图 5-21 搜索过程图

在用户搜索的时候，除了搜索到用户需要的内容，还会推荐一些相关的短片，短片数据根据表 5-6 获取。

序号	数据项	描述	页面打开方式
1.	链接	不展示	
2.	截图	截图可点击，点击新开页面打开链接	新开页面
3.	高清标志	若是高清片，则在截图左上角显示高清标志	
4.	片长	在截图左下角显示视频片长。格式为:分钟数+秒数，如“138:05”	
5.	标题	可点击，地址同截图。最多展示 14 个字加省略号	新开页面

表 5-6 短片显示所需数据

短片的表现方式如图 5-22。

图 5-22 短片显示图

而用户搜索到的视频显示数据则根据表 5-7.

序号	数据项	描述	连接打开方式
1.	链接	地址链接，不展示	新开页面
2.	截图	可点击，打开链接。	新开页面
3.	高清标志	若是高清片，则在截图左上角显示高清标志	
4.	片长	在截图左下角显示视频片长	
5.	标题	可点击，打开链接	新开页面
6.	来源	来源网站，如果有有中文名称就显示中文名称，否则显示主域名	
7.	发布时间	发布时间，不可点击	

表 5-7 视频显示所需数据

5.1.3数据删除模块

删除接口统一为 del.xml，del.xml 需要进行改版升级， 先要抓取平台上把删除的数据进行处理，放入队列，然后从队列中取数据，按照图 5-19 的流程调用。

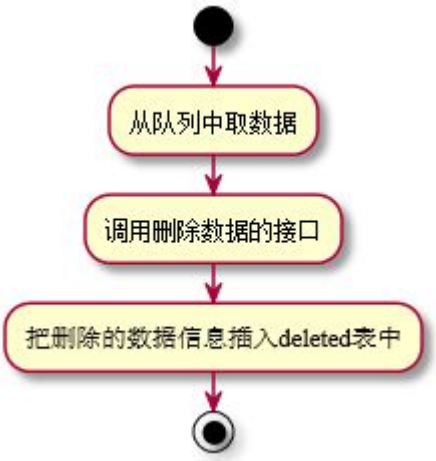

图 5-23

5.1.4数据归并模块

在数据抓取后，不能立即入库，需要先对数据进行处理，先计算 web_hot，然后从队列中取 gid 算 defultids(根据剧集和 web_hot，ps:条件是 videos 是没有被 forbidden 的，接着用 defaultids[0]的 web_hot，new_episode 分别更新 vgroups.web_hot，vgroups.now_episode，再针对人工对 hot_level 重新计算分数以及针对人工对 good_level 重新计算分数，ps:vgroups 的用于排序的三个分数，即:

（1）hot_level = web_hot + 人工 ＋ 百度 + 年份得分；

（2）web_hot = default video 的站内相对排名分（web_rank）*80 + vgroup 的入库时间分数*20；

（3）web_rank = 1 - M（以 hot 计算的站内相对排名）/N（站内总剧数） -> hot(当前剧情的 hot 值)/MAX_HOT（本站点的最好 hot 值)

vgroup 的入库时间分数 = 1 - M（入库时间倒序排名）/N(总数) 。简化为 vgroup 自增 id/总数 -> M(本剧入库时间)/N（最近的入库时间) new_level = ctime + 人工 good_level = 0.01*score + 豆瓣(<=9.8 时) + 人工。

5.1.5数据排序模块

在合作数据查询策略里，参与查询的字段包括名称、主演和导演，匹配的规则为索引字段包含查询词即返回结果。排序的规则如表 5-8 所示。

匹配优先级	匹配情况	说明	举例
1	搜索行为发生的页面类型	当条件 1 相同时，搜索行为发生页面类型的结果排在前面。 类型有电影、电视剧、动漫、综艺 如果搜索行为发生的页面不在这 4 条理，则认为该条件的权重相同	如在电视剧页面搜索“画皮”，同为精确匹配，电视剧“画皮“排在电影”画皮“前面
2	播放次数	当条件 2 相同时，按照播放次数倒排	如搜索“爱“，在所有部分命中的电视剧中，播放次数高的电视剧排在前面

表 5-8 排序规则

5.2 本章小结

本章对本影视导航网站的数据处理模块进行了详细的设计，同时介绍了相关前端界面的设计，针对各个模块给出了详细的设计和实现方案，通过类图、过程图和各类界面截图等直观的展示系统的构成。

6 系统测试

为了保障系统的正确性和功能的完整性，我们需要在特定的情况下，对各个功能进行测试，系统测试作为系统开发生命周期中的一个重要阶段，我们需要通过测试来确保系统的质量[v]。

6.1 功能性测试用例设计

首先根据规则制定测试用例，再根据测试用例设计测试用例表进行测试。在测试过程中，为了保障完整、无遗漏的测试，采用单元测试框架，根据 PHP 框架的源代码编写测试，使用判定、异常处理等表示单元测试可行。在进行测试时，要求测试目标明确：熟悉功能需求，要根据功能实现过程来设计输入和输出数据，要重视代码覆盖率和边界值覆盖，保证 Case 独立性。

由于功能过多，故选取两部分，进行测试说明。针对数据首次入库标题处理所设计的标准如图 6-1，豆瓣数据的匹配抓取设计的标准如图 6-2。

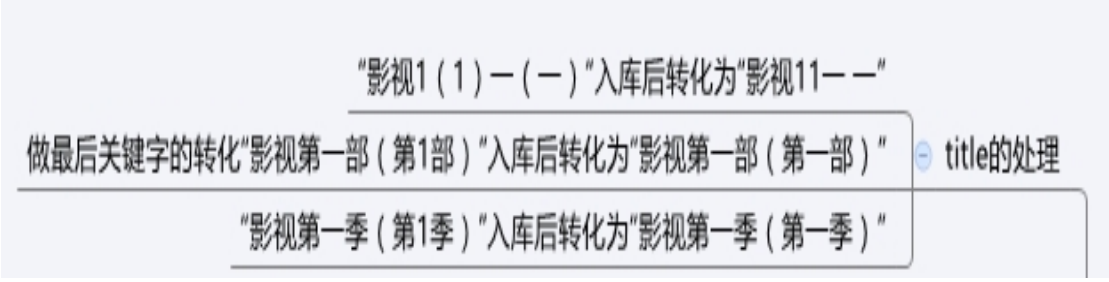

图 6-1 标题入库标准

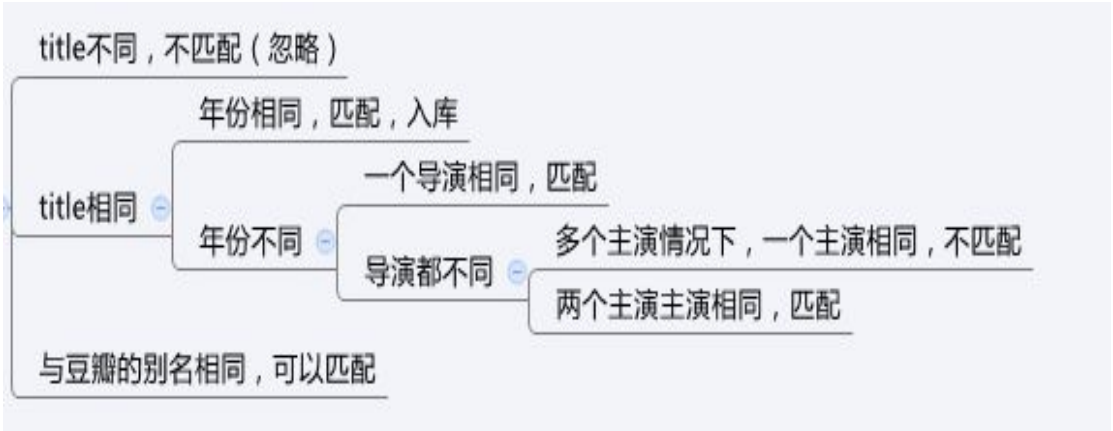

图 6-2 豆瓣数据匹配抓取标准

这里的测试标准是根据利用数据挖掘技术和正则表达式整合数据的特点制定的。测试的时候输入内容是根据抓取标准造数据，具体按照测试用例中的母节点，输出是调用phpunit 里的 assert 函数，按照测试用例里的最后一个子阶段输出。

本节主要根据图 6-1 和图 6-2 设计测试用例表，描述测试过程和结果，并加以说明。

用例编号	TC001
用例名称	数据入库对标题整合处理的准确性测试
测试人	胡琪乐
测试日期	2016-04-13
前置条件	无
测试步骤	（1） 对系统进行配置，启动系统并监控库内综艺数据 （2） 建立数据，数据内容为“极限挑战第 1 季”，让系统获取数据入库 （3） 监测系统对数据的处理，看系统能否将数据正确整合为“极限挑战第一季”再入库
预期结果	系统识别到 Case，并且成功将标题整合并显示

表 6-1 标题入库测试用例表

用例编号	TC002
用例名称	豆瓣数据抓取匹配的准确性测试
测试人	胡琪乐
测试日期	2016-04-13
前置条件	无
测试步骤	（1） 对系统进行配置，启动系统并监控库内豆瓣数据 （2） 造一个数据，数据内容为“北京遇上西雅图，2013，薛晓路，吴秀波”，让系统获取数据入库 （3） 监测系统的反应，看系统能否正确返回抓取信息
预期结果	（1） 系统能够识别出 Case， （2） 系统能够识别出 Case 的标题、年份、导演匹配 （3） 系统能够识别出多个主演，仅一个主演相同的情况下不予以匹配 （4） 系统能够判定该条 Case 不入库

表 6-2 豆瓣数据抓取匹配测试用例表

6.2 系统的非功能性测试

非功能测试是为了验证系统的可靠性、稳定性、轻便性、实用性等性能。本节主要通过一个测试用例表 6-3 来测试系统性能。

用例编号	TC003
用例名称	系统性能的测试
测试人	胡琪乐
测试日期	2016-04-013
前置条件	无
测试步骤	（1） 对系统进行配置，启动系统并同时监控库内电影、电视剧数据 （2） 同时输入 100 条电影、电视剧数据入库，数据中包含可以被识别的 Case 信息 （3） 查看系统的反应，看是否能够对每一条的视频数据都能够准确无误的进行抓取和判定
预期结果	系统能够识别出每一条视频数据中的 Case，并对每一条视频数据都进行抓取结果并成功反馈（入库 or 不入库）。

表 6-3 系统性能测试用例表

6.3 测试结果及分析

针对系统功能方面的测试，两个测试用例的结果都和预期的相同。测试用例表 1 中的标题数据被成功整合之后入库。测试用例表 2 中的豆瓣电影数据被成功匹配判定不符合要求不入库。

针对系统非功能方面的测试，测试用例表 3 中的运行结果和预期相同，每条数据都能成功抓取并识别，并且在该过程中没有出现崩溃、卡顿的情况，达到了使用的要求。

6.4 本章小结

本章从测试目标、测试方法和测试结果三个方面介绍了测试的过程，测试都已达到了预期结果，说明导航网站数据处理功能运行正常。

7 结论

随着互联网慢慢渗透进人们生活的方方面面，影视导航网站作为众多导航网站中的一类，受到了众多互联网人的重视和青睐，它在以其方便简洁快速的特点吸引众多网友访问增加网站访问量的同时，也能吸引广告商投资，为公司带来极大的利润。

本文主要介绍了影视导航网站在数据的处理部分的设计与实现，通过对需求的分析，进行了数据处理策略的设计，提出方案，并成功实现了数据的一系列处理。笔者的主要工作是参与数据处理策略的部分设计实现和后期测试，在实际项目中，笔者的系统设计与实现的能力得到了较大的提高。随着需求的日益增长，系统的功能将会随着需要逐步完善：

（1）由于在数据处理过程中需要创建多个实例，数据库经过长时间的累积存储数据之后，对维护会存在一定的困难。未来考虑优化系统平台，用以监测系统运行和实例的管理。

（2）网友在搜索视频的时候可能存在名称记不全且有错别字的情况，在检索视频数据的过程中就容易跳出很多无关视频，影响用户体验。未来考虑将检索功能设计的更加智能，也可以考虑改变检索的方式，更便于用户搜索。

（3）在处理数据的时候，制定的规则还存在遗漏，并且由于需要整合的类型过多，处理的速度还是不够快。未来考虑寻找能简化处理数据的方案。

参考文献

[1]王石，杨英娜. 精通 PHP+MySQL[M]. 北京：人民邮电出版社，2006.
[2] 高洛峰. 细说 PHP[M], 北京：电子工业出版社，2009.
[3] 胡军伟,秦奕青,张伟. 正则表达式在 Web 信息抽取中的应用[J]. 北京信息科技大学学报(自然科学版),2011,06:86-89.
[4]吴阿妹. 基于 MongoDB 的分布式搜索引擎技术研究[J]. 2014.
[5] 李宁,李战怀. 基于黑盒测试的软件测试策略研究与实践[J].计算机应用研究,2009.
[6]潘凯华,邹天思.PHP 开发实战宝典[M].北京:清华大学出版社,2010.
[7]施家庆.基于 PHP 的 SFF-MVC 框架研究[D].2007.
[8]张翼.Web 环境下数据库系统安全访问控制机制研究.2007.
[9]（美）萨莱 著，梁志敏，蔡建译，Professional PHP Design Patterns，PHP 设计模式，清华大学出版社，2010.

致　　谢

首先，十分感谢×××老师对我的毕业论文的指导，从实习，到毕业设计开题、初稿以及最终的定稿，张老师给了我很多的意见和帮助。张老师一丝不苟的工作风格对我影响很大。在论文的内容、思路、规范等方面都给了我很多的指导，非常感谢老师。

同时也要感谢公司同事对我的指导和帮助，让我将学到的知识运用到实习项目中，和新接触的知识融会贯通，使我受益匪浅。我不仅学到了新的知识，也更深刻的理解了团队合作的重要性，同时也学到了很多的工作经验，对我今后的学习和工作都有很大的帮助。

还要感谢北京交通大学的各位老师四年来对我的悉心教导，让我掌握了很多专业知识，找到自己的社会价值，也很谢谢家人对我的支持和关心，他们的关爱是我永恒的动力。

最后，感谢所有参与评阅笔者论文的各位老师，感谢你们严谨、细致的工作，我由衷的希望能得到你们的肯定，辛苦了。

附 录

翻译原文：

Extended Regular Expressions: Succinctnessand Decidability

By Dominik D. Freydenberger

Abstract ：Most modern implementations of regular expression engines allow the use of variables (also called backreferences). The resulting extended regular expressions(which, in the literature, are also called practical regular expressions, rewbr, or regex)are able to express non-regular languages.The present paper demonstrates that extended regular-expressions cannot be minimized。effectively (neither with respect to length, nor number of variables), and
that the tradeoff in size between extended and “classical” regular expressions is not bounded by any recursive function. In addition to this, we prove the undecidability of several decision problems (universality, regularity, and cofiniteness) for extended regular expressions.

Furthermore, we show that all these results hold even if the extendedregular expressions contain only a single variable.

Keywords：Extended regular expressions · Regex · Decidability · Non-recursive tradeoffs

1 Introduction

Since being introduced by Kleene [23] in 1956, regular expressions have developed
into a central device of theoretical and applied computer science. On one side, research
into the theoretical properties of regular expressions, in particular various aspects of their complexity, is still a very active area of investigation (see Holzer and Kutrib [20] for a survey with numerous recent references). On the other side, almost all modern programming languages offer regular expression matching in their A preliminary version of this article appeared as [15].D.D. Freydenberger () Institute for Computer Science, Goethe-University, Postfach 11 19 32, 60054 Frankfurt am Main。standard libraries or application frameworks, and most text editors allow the use of regular expressions for search and replacement functionality.But, due to practical considerations (cf. Friedl [16]), most modern matching engines have evolved to use an extension to regular expressions that allows the user to specify non-regular languages. In addition to the features of regular expressions as they are mostly studied in theory (which we, from now on, call proper regular expressions), and apart from the (regularity preserving) “syntactic sugar” that
most implementations use, these extended regular expressions contain backreferences, also called variables, which specify repetitions that increase the expressive power beyond the class of regular languages. For example, the (non-regular) language

L = {ww | w ∈ {a,b}* } is generated by the extended regular expression α := ((a | b)*)%x x。This expression can be understood as follows (for a more formal treatment, see Definition 4): For any expression β, (β)%x matches the same expression as β, and binds the match to the variable x. In the case of this example, the subexpression((a | b)*)%x can be matched to any word w ∈ {a,b}* , and when it is matched to w, the variable x is assigned the value w. Any further occurrence of x repeats w,leading to the language of all words of the form ww with w ∈ {a,b}* . Analogously,the expression ((a | b)*)%x xx generates the language of all www with w ∈ {a,b}* .

Although this ability to specify repetitions is used in almost every modern matchingengine (e.g., the programming languages PERL and Python), the implementationsdiffer in various details, even between two versions of the same implementation of a programming language (for some examples, see Câmpeanu and Santean [7]).Nonetheless, there is a common core to these variants, which was first formalized by Aho [2]. Later, Câmpeanu et al. [9] introduced a different formalization that is closer to the real world syntax, which addresses some questions of semantics that were implicitly left open in [2]. In addition to this, the pattern expressions by Câmpeanu and Yu [8] and the H-expressions by Bordihn et al. [5] use comparable repetition mechanisms and possess similar expressive power. Still, theoretical investigation of extended regular expressions has been comparatively rare (in particular when compared to their more prominent subclass); see e.g., Larsen [25], Della Penna et al. [14], Câmpeanu and Santean [7], Carle and Narendran[10], and Reidenbach and Schmid [30].

In contrast to their widespread use in various applications, extended regular expressions have some undesirable properties. Most importantly, their membership problem (the question whether an expression matches a word) is NP-complete (cf.Aho [2]); the exponential part in the best known upper bounds depends on the number of different variables in the expression. Of course, this compares unfavorablyto the efficiently decidable membership problem of proper regular expressions (cf.Aho [2]). On the other hand, there are cases where extended regular expressions express regular languages far more succinctly than proper regular expressions. Consider the following example:

Example 1 For $n \geq 1$, let $L_n := \{www \mid w \in \{a,b\}+, |w| = n\}$.

These languages L_n are finite, and hence, regular. For the sake of this example, we define the length of an extended regular expression α as the total number of symbols that occur in α (in literature, this measure is often called size, cf. [21]).

Theory Comput Syst (2013) 53:159–193 161With some effort, one can prove that every proper regular expression for L_n is atleast of length exponential in n, e.g., by using the technique by Glaister and Shallit[17] to prove that every NFA for L_n requires at least $O(2n)$ states. Due to the construction used in the proof of Theorem 2.3 in Hopcroft and Ullman [22], this alsogives a lower bound on the length of the regular expressions for L_n.In contrast to this, L_n is generated by the extended regular expression$\alpha_n := ((a \mid b) \cdot \cdot \cdot (a \mid b)$ n times $(a \mid b))\%x\ xx$,which is of a lengththat is linear in n.

Due to the repetitive nature of the words of languages L_n in Example 1, it is not surprising that the use of variables provides a shorter description of L_n. The followingexample might be considered less straightforward:

Example 2 Consider the expression $\alpha := (a \mid b)*\ ((a \mid b)+)\%x\ x(a \mid b)*$.

It is a wellknown fact that every word $w \in \{a,b\}*$ with $|w| \geq 4$ can be expressed in the form $w = uxxv$, with $u, v \in \{a,b\}*$ and $x \in \{a,b\}+$ (as is easily verified by examining all four letter words). Thus, the expression α matches all but finitely many words; hence,its language $L(\alpha)$ is regular.

Example 2 demonstrates that the use of variables can lead to languages that are (non-trivially) regular. The phenomenon that an expression like α can generate a cofinite language is strongly related to the notion of avoidable patterns (cf. Cassaigne[11]), and involves some very hard combinatorial questions (in particular, Example2 illustrates this connection for the pattern xx over a binary alphabet).We observe that extended regular expressions can be used to express regular languagesmore succinctly than proper regular expressions do, and that it might be hardto convert an extended regular expression into a proper regular expression for thesame language.The two central questions studied in the present paper are as follows: First, how

hard is it to minimize extended regular expressions (both with respect to their length,and with respect to the

number of variables they contain), and second, how succinctlycan extended regular expressions describe regular languages? These natural questions
are also motivated by practical concerns: If a given application reuses an expression many times, it might pay off to invest resources in the search for an expression that is shorter, or uses fewer variables, and thus can be matched more efficiently.
We approach this question through related decidability problems (e.g., the universality problem) and by studying lower bounds on the tradeoff between the size of extended regular expressions and proper regular expressions.

The main contribution of the present paper is the proof that all these decision problems are undecidable (some are not even semi-decidable), even for extended regular expressions that use only a single variable. Thus, while bounding the number of variables in extended regular expressions (or, more precisely, the number of variable bindings) reduces the complexity of the membership problem from NP-complete to 162 Theory Comput Syst (2013) 53:159–193polynomial (cf. Aho [2]), we show that extending proper regular expressions with only a single variable already results in undecidability of various problems.

As a consequence, extended regular expressions cannot be minimized effectively, and the tradeoff between extended and proper regular expressions is not bounded by any recursive function (a so-called non-recursive tradeoff, cf. Sect. 2.3 for additional context). Thus, although the use of the “right” extended regular expression for a regular expression might offer arbitrary advantages in size (and, hence, parsing speed), these optimal expressions cannot be found effectively. These results highlight the power of the variable mechanism, and demonstrate that different restrictions than the number of variables ought to be considered.
The structure of the further parts of the paper is as follows: In Sect. 2, we introduce most of the technical preliminaries. Section 3 consists of Theorem 10 (the main undecidability result), its proof, and the required additional technical preliminaries, while Sect. 4 discusses the consequences and some extensions of Theorem 10.

2 Preliminaries

This paper is largely self-contained. Unexplained notions can be found in Hopcroft and Ullman [22], Cutland [13], and Minsky [27].

2.1 Basic Definitions

Let N be the set of natural numbers, including 0. The function div denotes integer division, and mod denotes its remainder (e.g., 5 div 3 = 1 and 5mod 3 = 2). The symbol∞ denotes infinity. The symbols $\subseteq$, $\subset$, $\supseteq$ and $\supset$ refer to the subset, proper subset, superset and proper superset relation, respectively. The symbol $\varnothing$ denotes the empty set, \ denotes the set difference (defined by $A \setminus B := \{x \in A \mid x \notin B\}$). For every set A, P(A) denotes the power set of A.We denote the empty string by λ. For the concatenation of two strings w1 and w2, we write $w1 \cdot w2$ or simply w1w2. We say a string $v \in A^*$ is a factor of a string $w \in A^*$ if there are $u1,u2 \in A^*$ such that w = u1vu2. The notation |K| stands for thesize of a set K or the length of a string K.If A is an alphabet, a (one-sided) infinite word over A is an infinite sequencew = $(wi)_{i=0}^{\infty}$ with $wi \in A$ for every $i \geq 0$. We denote the set of all one-sided infinitewords over A by A^ω and, for every $a \in A$, let a^ω denote the word w = $(wi)_{i=0}^{\infty}$with wi = a for every $i \geq 0$. We shall only deal with infinite words $w \in A^\omega$ that have the form $w = u\,a^\omega$ with $u \in A^*$ and $a \in A$. Concatenation of words and infinite words is defined canonically: For every $u \in A^*$ and every v

∈ Aω with v = (vi)∞i=0,
u · v := w ∈ Aω, where w0 · · · · ·w|u|−1 = u and wi+|u| = vi for every i ≥ 0, while vu is undefined. In particular, note that a aω = aω for every a ∈ A.

2.2 Extended Regular Expressions

We now introduce syntax and semantics of extended regular expressions. Apart from some changes in terminology, the following definition of syntax is due to Aho [2]:Theory Comput Syst (2013) 53:159–193 163Definition 3 Let Σ be an infinite set of terminals, let X be an infinite set of variables,and let the set of metacharacters consist of λ, (,), |, ∗ , and %, where all three sets arepairwise disjoint.We define the set of extended regular expressions to be the smallest set that satisfies the following conditions:

1. Every a ∈ Σ ∪ X ∪ {λ} is an extended regular expression.
2. If α1 and α2 are extended regular expression, then
(a) (α1)(α2) (concatenation),
(b) (α1) | (α2) (alternation),
(c) (α1)∗ (Kleene star)are extended regular expressions.
3. For every extended regular expression α and every variable x ∈ X such that %x is not a factor of α, (α)%x is an extended regular expression (variable binding).

We denote the set of all extended regular expressions by RegEx. A proper regular expression is an extended regular expression that contains neither %, nor any variable (hence, proper regular expressions are those expressions that are commonly called “regular expressions” in theoretical computer science).If an extended regular expression β is a factor of an extended regular expressionα, we call β a subexpression of α. We denote the set of all subexpressions of α bySUB(α).

We shall use the notation (α)+ as a shorthand for α(α)∗ , and freely omit parentheses whenever the meaning remains unambiguous. When doing this, we assume that there is a precedence on the order of the applications of operations, with ∗ and +ranking over concatenation ranking over the alternation operator |.

In Aho [2], an informal definition of the semantics of extended regular expressions is given. In Aho’s approach, extended regular expressions are interpreted as language generators in the following way: An extended regular expression α is interpreted from left to right. A subexpression of the form (β)%x generates the same language as the expression β; in addition to this, the variable x is bound to the word w that was generated from β (if x already has a value, that value is overwritten). Every occurrence of x that is not in the context of a variable binding is then replaced with w. When following this approach, there are some cases where the semantics are underspecified. For example, Aho [2] does not explicitly address the rebinding of variables (cf.

Example 5, further down), and the semantics of expressions like ((a)%x | b)x are unclear. Although the proofs in the present paper are not affected by the ambiguities that arise from the informal approach, we include a formal definition of the semantics, which is an adaption of the semantics of Câmpeanu et al. [9] to the syntax from Definition 3.

外文文献翻译：

扩展正则表达式:简洁和可判定性

作者：Dominik D. Freydenberger

摘要：大多数现代实现正则表达式的引擎允许使用 变量（也称为反向引用）。他由此得出结论扩展正则表达式（其中，在文献中，也称为实用的正则表达式）是能够表达非常规语言的。本文将表明,扩展正则表达式不能有效的被最小化(对长度和数量的变量)，并且在扩展和“经典”正则表达式之间的大小的折衷不是有任何递归函数有界。此外，我们还会证明不可判定性几个决策问题（普遍性、规律性，而且还有）来扩展正则表达式。此外，我们还会证明，即使扩展正则表达式只包含一个变量，所有这些结果依然成立。

关键词：扩展正则表达式，正则表达式，判定，非递归权衡

1.引言

自 1956 年被介绍克莱尼提出之后，正则表达式已经发展到了中央装置的理论和计算机科学的应用中。一方面，研究理论属性的正则表达式，特别是到各个方面它们的复杂性，仍是非常活跃的领域的调查 (见霍尔和 Kutrib [20] 与众多近期文献调查)。另一边，几乎所有现代编程语言的他们文章的中提供了匹配正则表达式的初稿。

标准库或应用程序框架和大多数文本编辑器允许使用正则表达式搜索和替换功能。但是，基于实际的考虑 （参看市况 [16]），大多数现代匹配引擎进化成了对正则表达式的扩展，并允许用户使用指定非常规语言。除了正则表达式的功能，由于它们大多是基于理论上的 （于是我们，从现在开始，调用适当的定期表达式），且除了 （规律保留）"语法糖"，大多数实现使用，这些扩展的正则表达式包含反向引用，也称为变量，指定重复增加的表现力
超越类的正规语言的力量。例如，（非常规） 语言 L = {ww | w ∈ {a，b} ＊ } 由扩展正则表达式Ａ: = ((a | b)＊) %x x。此表达式可以理解如下 （更正式的处理，请参见定义 4） ： 任何表达 β，（β） %x 火柴 β，相同的表达和绑定到变量 x 的这场比赛。在此示例中，子表达式((a | b)＊) 可以与任何单词匹配 %x w ∈ {a，b} ＊ ，并且当它符合 w，变量 x 被分配值 w。X 任何进一步发生重复 w，类似地，关于语言的所有单词的形式 ww w ∈ {a，b}？ 。该表达式 ((a | b)＊ %X xx？) 生成了所有 www 与 w ∈ {a，b} ＊ 的语言。

虽然这能够指定重复使用在几乎每一个现代匹配引擎(如。PERL 和 Python 编程语言),实现两个版本之间的不同在不同的细节,甚至相同的实现的编程语言(一些例子,看到 Campeanu Santean[7])。但这是第一次正式的使这种变异成为常见的核心[2]。后来,Campeanu et al。[9]引入了不同的规范化,更接近现实世界的语法,这地址有些问题隐含的语义在[2]。此外,通过 Campeanu 和模式表达式于[8]和 H-expressions Bordihn 等。[5]使用重复类似的机制
和拥有类似的表达能力。不过,理论研究相对扩展正则表达式是罕见的(尤其是相比更加突出子类);看到如拉森[25],德拉 Penna et al。[14],Campeanu 和 Santean[7],卡尔和 Narendran[10],Reidenbach 施密德[30]。在被广泛使用在各种应用程序中,扩展正则表达式
有一些不良的特性。最重要的是,他们 membershipproblem(质疑一个表达式匹配一个单词)是 np 完备性(cf。阿霍[2]);指数中最著名的上界取决于数量不同变量的表达式。当然,这比
有效的可决定的成员正确的正则表达式(cf 的问题。阿霍[2])。另一方面,在某些情况下,扩展正则表达式表达正则语言简洁地远远超过适当的正则表达式。参考下面的例子:

示例 1(n≥1,让 Ln:= { www | w∈{ a、b } +,| | = n }。这些语言 Ln 是有限的,因此,常规。为了这个例子中,我们定义一个扩展正则表达式的长度α总数发生在α(这种方法通常被称为大小、cf。[21])。

通过一定的努力，可以证明每一个适当的正则表达式为 Ln 是在最小长度指数 n，例如，通过使用由莱斯特和 Shallit 技术的[17] 证明每个 NFA 的 Ln 至少需要 O(2n) 国家。这也证明定理 2.3 Hopcroft 和厄尔曼 [22] 中使用的施工给出了下限巷的正则表达式的长度与此相反，Ln 生成的扩展正则表达式α n := ((a | b) ・ ・ ・(| b）。n 次 (| b）。) %x xx，这是中 n 是线性长度。由于单词的语言的示例 1 中 Ln 的重复性质，它不是令人惊讶的是，变量的使用提供了较短的说明的巷以下示例可能被认为不那么简单 ：示例 2 考虑表达 α : = (| b）。＊ ((a | b)+) %x x(a | b)？.这是家喻户晓的。

事实上，每一个字 w ∈ {a，b} ＊ 与 |w|≥ 4 都可以表示在窗体中 w = uxxv， v ∈ {a，b} ＊ 和 x ∈ {a，b} +（如通过检查所有容易验证四个字母的单词）。因此，表达 α 匹配所有但有限多字;因此，它的语言 L(α) 是定期的。

示例 2 说明变量的使用会导致语言（非-琐细） 定期。这样 α 的表达式可以生成的现象 cofinite 语言密切相关的概念可以避免模式 （参见 Cassaigne[11]），和涉及一些很难组合问题 （特别是，示例 2 说明了此连接模式 xx 二进制字母表）。我们观察扩展正则表达式可以用于表达正则语言比做适当的正则表达式，它可能很难更简洁将扩展的正则表达式转化为适当的正则表达式同一种语言。

本论文研究的两个核心问题是：第一，最大限度地减少扩展正则表达式有多难？（两个相对于它们的长度，相对于变量所包含的号码），第二，如何简洁可以扩展正则表达式来描述正则语言吗？这些自然的问题也是出于实际的考虑：如果一个给定的应用程序使用一个表达式很多时候，它可能会支付的投资资源，寻找一个表达式，即较短，或使用较少的变量，从而可以更有效地匹配。我们研究这个问题，通过相关的可判定性问题（例如，普遍性问题）和通过研究下界之间的权衡的大小扩展正则表达式和正则表达式。本文的主要贡献是证明，所有这些决策问题是不可判定的（有些甚至没有半可判定的），甚至扩展规则仅使用一个变量的表达式。因此，在边界的变量的数目在扩展正则表达式（或更确切地说，变量的数目
绑定）降低从 NP-完全到 162 理论计算系统（2013）53:159–193 多项式（参见阿霍[2]），我们发现，适当的扩展正则表达式只有一个单一的变量已经导致各种问题的不可判定性。
因此，扩展正则表达式不能有效地减少，扩展和适当的正则表达式之间的折衷是不有界的
任何递归函数（所谓的非递归的权衡，参见第。2.3 额外的语境）。因此，虽然“正确”的使用扩展正则表达式的规则表达式可以提供任意大小的优点（因此，解析速度），不能有效地找到这些最佳表达式。这些结果突出变量机构的功率，并表明不同的限制比应考虑的变量数。
本论文的结构如下：第 2 节，我们会介绍大部分的技术准备工作。第 3 节由定理 10（主要说明不可判定的结果），它的证明，以及所需的额外的技术准备工作。第 4 节讨论了定理 10 的结果和一些扩展。

2.前言

本文在很大程度上是独立的。其中无法解释的概念可以 Hopcroft 中找到 Ullman[22],Cutland[13],和明斯基[27]。

2.1 基本定义

使 N 代表自然数的集合,包括 0。函数 div 表示整数的除法,它的模表示其剩余部分(如。5 div 3 = 1 和 5||3 = 2)。符号∞表示无穷。符号⊆，⊂，⊇和⊃指的子集，子集，超集和适当的超集关系，分别。符号∅ 表示空集，\表示集合差（由 A、B：= { x∈一| X /∈b }）。对于每一个设置，P(A) 表示 A 的幂集。

我们用 λ 表示空字符串。连接两个字符串的 w1 和 w2,我们写 w1 • w2 或者只是 w1w2。我们说 v∈字符串＊ 是一个因素的一个字符串 w∈＊ 如果有 u1,u2∈＊ 这样 w = u1vu2。符号| | K 代表大小设置 K 或一个字符串的长度 K。如果是一个字母,一个(单边)无限的词是一个无穷序列 w =(wi)∞i= 0 与 wi∈每≥0。我们表示所有片面无限的集合话说 A ω 和,每一个∈,让 A ω 表示这个词 w =(wi)∞i= 0,wi =所有的 i≥0。我们只处理无限词汇 w∈A ω 有形式和 u∈w = u a ω 吗＊ ∈a 连接的单词和无限的词语定义:每一个 u∈＊ 每个 v∈A ω 与 v =(vi)∞i= 0,u • v:= w∈A ω ,w0 u • • • • • w | |−1 = u 和 wi + | | = vi 所有的 I≥0,而 vu 是未定义的。特别是,a ω =每一个∈a ω 。

2.2 扩展正则表达式

我们现在介绍扩展正则表达式的语法和语义。除了有些术语的变化,以下定义的语法是由于阿霍[2]:第一版系统理论(2013)53:159 193 - 193 定义 3 让 Σ 无限的终端,让 X 是一个无限的变量,,让元字符的集合包括 λ ,(,),|,＊ ,和%,所有三组两两不相交。我们定义的集合扩展正则表达式是最小的满足下列

条件:

1.每一个∈ Σ ∪X∪ { λ }是一个扩展正则表达式。

2 如果 α 1 α 2 扩展正则表达式,那么

(a)(α 1)(α 2)(连接),

(b)(α 1)|(α 2)(变更),

(c)(α 1)* (克林星号)是扩展正则表达式。

3.每一个扩展的正则表达式 α 每个变量 x x，x 是∈不是一个因子，（1）（1）是一个扩展正则表达式（变量绑定）。

我们表示所有扩展正则表达式正则表达式的设置。一个适当的规则表达式是一个扩展正则表达式，它包含两个变量，也不包含任何变量（因此，正则表达式是通常被称为的表达式
理论计算机科学中的“正则表达式”。如果扩展正则表达式是一个扩展正则表达式的一个因素 α，我们称 β 子表达式 α 。我们表示所有的子表达式。我们将使用符号（“+”）作为一个缩写为（1）＊，随意省略括号无论何时意义仍不明确。当这样做时，我们假设那是在业务应用的顺序优先，有＊ +排名在级联排序选择操作符|。

在阿霍中[2]，对扩展正则表达式的语义给出了一个非正式的定义。在以前的方法中，扩展的正则表达式解释为语言在下列方式中的发电机：一个扩展的正则表达式的表达式被解释从左至右。子表达式的形式（ β ）%×产生相同的语言作为表达式的表达式，除了这个外，该变量的*也被绑定到了这个词产生的 β （如果 X 已经有了价值，价值是覆盖）。每一在一个变量绑定的上下文中不在发生的情况，然后用 W 替换。当遵循这种方法，有一些情况下的语义是指定的。例如，阿霍[2]不明确地址的绑定变量（参见例 5，进一步下跌），和喜欢的语义表达（（P(A)）%×| B）X 尚不清楚。虽然在本文件中的证明没有受到影响的含糊之处，但是从一些非正式的方法中，我们包括了一个正式的语义定义，这是 Câmpeanu 等人的语义的一种适应。[9]的语法参见定义 3。