|  |  |
| --- | --- |
| 单位代码 | 10006 |
| 学号 | 10061167 |
| 分类号 | TP391 |



****

毕业设计(论文)

毕业论文格式检查系统的设计与实现

|  |  |
| --- | --- |
| 院（系）名称 | 计算机学院 |
| 专业名称 | 计算机科学与技术 |
| 学生姓名 | 刘牛顿 |
| 指导教师 | 李莹 |

**2015**年**6**月

**毕业论文格式检查系统的设计与实现** 刘牛顿 北京航空航天大学

北京航空航天大学

**本科毕业设计（论文）任务书**

Ⅰ、毕业设计（论文）题目：

毕业论文格式检查系统的设计与实现

Ⅱ、毕业设计（论文）使用的原始资料（数据）及设计技术要求：

本文的设计技术要求：设计并实现一个针对本科生毕业论文的格式检查系统，能够根据具体的格式模版的要求，实现对学生论文中出现的格式错误进行比对查找，并生成错误报告提交，作后续的进一步处理工作。该系统脚本使用python语言完成，具有良好的封装性，提供命令行接口，使之能方便其他程序调用。

Ⅲ、毕业设计（论文）工作内容：

1、研究xml标签定义、word文本格式提取和定位的实现方法；

2、使用python及相关技术实现本系统；

3、设计系统测试方案，检测系统是否符合设计要求；

Ⅳ、主要参考资料：

[1] Virantha Ekanayake，Reading and writing Microsoft Word docx files with Python，[J]Posted By Virantha On Fri 16 August 2013.

[2] Magnus Lie Hetland,Beginning Python From Novice to Professional,[M]人民邮电出版社，2014.6.

[3] 张印，基于VSTO的高校毕业论文格式化处理软件的设计与实现，[D]北京邮电大学，2012.

计算机 学院（系） 计算机科学与技术 专业类 110616 班

学生 刘牛顿

毕业设计（论文）时间： 2015年 3 月 1 日至 2015 年 6月 10日

答辩时间： 2015 年 6 月 日

成 绩：

指导教师： 李莹

兼职教师或答疑教师（并指出所负责部分）：

系（教研室） 主任（签字）：

本人声明

我声明，本论文及其研究工作是由本人在导师指导下独立完成的，在完成论文时所利用的一切资料均已在参考文献中列出。

作者：刘牛顿

签字：

时间：2015年6月

毕业论文格式检查系统的设计与实现

学 生：刘牛顿

指导教师：李 莹

摘 要

毕业论文的撰写是每一个大学生毕业前非常重要的一个环节，由于学术论文本身的严谨性和科学性，因此对论文格式的检查要求十分严格，采用程序自动检查格式的方法能够减少以往人工检查方式的耗时耗力。针对此需求，本文研究了解析word文档xml数据结构的相关技术以及类似的论文格式检查的实现方法，在此基础上设计了一个可行的论文格式检查系统，使用python语言进行实现，并且封装成带命令行参数的可执行程序，方便格式修改系统来进行调用分析。本系统针对本科学生毕业论文的格式模版，能够检查出学生论文中的字体属性、段落属性等基础格式错误和不符合撰写规范的广义格式错误，并生成错误报告提交给论文格式修改系统作进一步处理。此外，本文使用了一些存在常见问题的的论文样例对系统功能进行了测试，展示了本系统在实际执行过程中的效果。

关键词：数据挖掘，计算机思维导论，成绩分析系统

**A format check system designed and implemented for the graduation thesis**

Author: LIU Niu-dun

Tutor: LI Ying

**Abstract**

Writing the graduation thesis is a very important link for each college students before graduation. Due to academic rigor and scientificness of itself, so the paper format is very strict inspection, the program automatically check format method can reduce the time-consuming manual inspection way past.Aiming at this requirement, this paper studies the analytic technology related to word document XML data structure and the realization method of similar paper format to check on the basis of design a feasible thesis format inspection system, using the python language implementation, and encapsulated into an executable program with the command line parameters, convenient format modification analysis of system to make calls.This system for the undergraduate students of the graduation thesis format templates, students can check out the font properties, paragraphs, etc of the paper basic format and writing specification does not comply with the general format errors, and generate the error report submitted to paper format modification system for further processing.In addition, this paper USES some common problems existing in the paper the sample system function was tested, and shows the effect in the process of this system in the actual execution.

**Keywords:** format check, graduation thesis, xml analysis

目录

[1 绪论 1](#_Toc421723981)

[1.1 研究背景与意义 1](#_Toc421723982)

[1.2 国内外研究现状 2](#_Toc421723983)

[1.3 研究目标和主要工作 3](#_Toc421723984)

[1.4 论文结构安排 4](#_Toc421723985)

[2 相关技术 5](#_Toc421723986)

[2.1 python概述 5](#_Toc421723987)

[2.2 lxml类库 7](#_Toc421723988)

[2.3 word文档的xml结构标签定义 8](#_Toc421723989)

[2.4 pyinstaller工具 9](#_Toc421723990)

[3 系统设计 11](#_Toc421723991)

[3.1系统构架 11](#_Toc421723992)

[3.2 系统模块 12](#_Toc421723993)

[3.2.1 定制文档规则 12](#_Toc421723994)

[3.2.2 文本提取和格式比对 14](#_Toc421723995)

[3.2.3 检查结果反馈 16](#_Toc421723996)

[3.3 关键技术解决方案 16](#_Toc421723997)

[4 系统实现 19](#_Toc421723998)

[4.1 系统整体运行过程 19](#_Toc421723999)

[4.2 读取xml模块 19](#_Toc421724000)

[4.3 获取节点包含的文本及格式信息 20](#_Toc421724001)

[4.4 定位论文内容 26](#_Toc421724002)

[4.5 检查并生成报告 29](#_Toc421724003)

[4.6 其他模块 32](#_Toc421724004)

[5 系统测试 33](#_Toc421724005)

[5.1 测试环境 33](#_Toc421724006)

[5.2 系统功能测试 33](#_Toc421724007)

[5.2.1 格式提取功能 33](#_Toc421724008)

[5.2.2 定位论文内容功能 35](#_Toc421724009)

[5.3 系统性能测试 36](#_Toc421724010)

[结论 38](#_Toc421724011)

[致谢 39](#_Toc421724012)

[参考文献 40](#_Toc421724013)

# 1 绪论

本章将介绍课题的选题背景和研究意义，分析现有的若干论文格式检查系统各自的特点，并阐述本文主要的研究内容以及论文结构安排。

## 1.1 研究背景与意义

毕业论文对高等院校本科专业教学计划来说是一个重要的组成部分，也是现在大多数学校对学生在校学习成果进行检查的一种重要措施。学校通过让学生撰写毕业论文这一任务，检查学生在校最后阶段的学习质量，同时根据发现的问题反馈，提高教学质量。所以毕业论文的重要性不言而喻，其意义影响深远，体现在两方面：

每一个大学生在大学学习的最后阶段，必须完成毕业论文的撰写任务，在申请学位时需要向学校提交相应的学位论文，经答辩通过后，才能取得学位，顺利毕业。对大学生而言，毕业论文是个人综合能力的显露和体现，是走向社会前检验在大学生活中所学知识和技能的巩固性与熟练度。每个学生在毕业设计中都要花费半个月至一个月的时间去完成毕业论文的撰写，因此可见毕业论文在毕设环节中所占的比重相当高。

同时对学校来说，毕业设计论文也是提高教学质量的重要环节，是学校保证人才素质的一种重要的措施。学校通过对毕业论文的严格审查，来提高学生在实践中发现问题、提出问题、独立分析的能力，提高学生综合运用其所学专业知识、理论基础来解决问题的能力，树立学生严肃认真的科学态度和理论联系实际的工作作风，也锻炼了学生行社会调查研究、收集和使用文献资料、综合论证、总结写作等基本技能在实践中的应用。学校自然会加大对毕业论文检查的力度。

由于其重要性，在各个学校毕业设计的环节中，毕业论文对学生们和老师来说都是一个相当耗费的时间、精力的一项工作。又由于学术论文本身的严谨性和科学性，在论文的写作过程中，格式的要求非常严格，不能有丝毫差错。在检查格式时，目前国内大多数高校仍采用人工检查的方式，检查工作繁重耗时耗力，容易出错[1]。在这个背景下，设计一个能自动检查毕业论文格式的系统十分必要，其意义主要有以下几个方面：

1）减少学生在论文格式校对上花费的时间和精力。如果学生在写作论文的过程中，还要把相当多的时间耗费在对论文格式的调整和校对中，那么本应真正花费在论文内容和质量方面的时间就会大大减少，从而导致论文质量的降低。

2）减轻指导老师在审阅论文时纠正格式错误的问题。指导老师在审查学生提交的毕业论文时，如果发现论文在格式上存在错误，就会耗费大量时间指导学生去修改论文的格式使之符合规范和要求，因为论文格式的正确性很大程度体现出学生的学习态度，影响了最终上交稿的合格率。这样的话老师花费在论文的研究价值性、内容的科学性、前瞻性方面的审查的时间就会大大缩少，老师在指导论文时往往会出现“重格式，轻内容”现象[2]。这种情况，最终导致学生的毕业论文水平总体降低，其内容的科学性、前瞻性以及研究意义等方面也会存在问题，这将大大地降低教学质量。

3）减少因格式出现错误而重新打印论文造成的纸张浪费现象。这一点也很好的体现了绿色环保的概念。

## 1.2 国内外研究现状

论文格式检查系统的核心在于解析word文档中的XML文件，即对word文档进行处理。国外论文看重内容的质量而很少对格式有硬性要求，所以国外对于格式检查的研究主要是基于word文档的处理工作。但对于word处理技术比较单一并且在效率、处理效果上与人工校正还存在一定的差距，各方面开展较为缓慢，造成此种情况主要由以下的原因：

1）软件企业开发较少。一般情况下企业在对工作内容、工作汇报管理时使用基于excel表格的数据处理较多，而基于word文字的应用相对较少，从而在对word格式处理和内容分析上面需求也相应减少，这就造成了大部分软件企业不大重视word处理技术等方面的技术研发，该方面技术发展相对来说也就比较滞后了[3]。

2）官方文档公开较少。微软发布Office相关技术时对于技术开发文档一直以来公布较少，并且主要还是面向用户来发布操作方法，对于程序分析人员很少关注。直到微软推出office 2007时，对于面向word对象的程序开发人员来说，才有了基于word对象操作的SDK和文档这样一份完善的技术方法。

国内则由于对毕业论文的重视，随着大学扩招，各大高校每年毕业的学生人数都在不断增加，而毕业论文又是每个学生毕业前的一道必修功课，那么必然导致在论文检查方面的工作量庞大且持续增加，为了解决这一问题，目前已经有部分院校出现对自动检查论文格式系统的研究，同时在市场上还有一些开源项目的产品。他们实现的方式各有不同，并且都有各自的特点。

比如北京邮电大学做的基于VSTO的毕业论文格式化处理系统，采用B/S与C/S相结合的设计模式,为用户提供论文标准化格式处理的网络工具，将处理完成的文档以邮件的形式发送回提交格式处理申请的用户的邮箱；华南农业大学信息学院做的基于JAVA的文档格式检查技术的应用，通过逐行扫描快速检查出错误输出结果，并对word文档的解压缩、读取格式等关键技术点做了较为详细的说明，但在模版设置上稍有欠缺；上海海事大学信息工程学院做的基于.NET的论文格式检查系统，采用三层分层结构，通过把毕业论文模版的段落分为若干类的方法来定位，给出给定模版文档和检测文档之间的格式差异，但如果错误信息集中时批注标示在视觉上欠佳[5]。

综上所述决定设计基于python的论文格式检查系统，使用python语言编写程序作为后台调用脚本来实现对论文格式的检查功能，并把检查结果生成报告反馈给前台作进一步处理。未来在格式处理这一方面的研究方向主要是能够让格式检查系统在对毕业论文进行格式检查工作时更加智能、信息化，减少人工操作产生的失误，大大提高毕业论文格式检查的效率，并在这个基础上可以将该技术推广至对任意word 文档的格式检查。

## 1.3 研究目标和主要工作

本文的研究目标是设计并使用python语言实现一个作为后台调用的论文格式检查系统，该系统能够根据目前本科生毕业论文书写的规则模版的要求，检查出论文中段落文本在格式上出现的错误。

本文完成的主要工作有以下几个方面：

1. 研究了现有论文格式检查系统的设计思路
2. 研究了word文档xml数据结构的标签定义
3. 研究了python实现解析xml的方法
4. 设计了可行的算法从xml中提取文本对应的格式，并根据文本定位判断其在论文中的位置
5. 使用测试样例展示了本系统在实际应用中的效果

## 1.4 论文结构安排

本文的主要内容分为以下章节：

第一章是“绪论”，介绍了课题的选题背景与意义，分析现有的论文格式检查系统各自的特点，并阐述本文的研究目标。

第二章是“相关技术”，详细介绍了本系统的设计与实现中使用的基础技术。

第三章是“系统设计”，介绍了本系统的设计思路，分析了设计中要解决的关键问题，并提出了对应的解决方案。

第四章是“系统实现”，分别介绍了系统的各个模块的实现方法，并针对实现中需要关键问题的做了详细介绍；

第五章是“系统测试”，针对系统的功能性进行了测试，并采用一些测试样例，用于展示本系统的实际使用效果。

第六章是“总结”，总结整个系统的设计与实现过程，分析本系统目前存在的不足之处，并提出简要的改进方案。

# 2 相关技术

本章将介绍系统在设计实现过程中学习和借鉴的相关技术思路。 本系统主要使用的工具或涉及到的技术有python编程语言、lxml类库、xml标签定义、pyinstaller工具。

## 2.1 python概述

Python是一种解释型的、面向对象的、带有动态语义的高级程序设计语言。Python语法简洁而清晰，具有丰富和强大的类库。自从Guido van Rossum在20世纪90年代初创造这门语言后，它的使用者就一直在稳步增加，而且近些年来，社区对它的兴趣也日益浓厚[12]。据统计，自从2004年以后，python的使用率呈线性增长，已经成为最受欢迎的程序设计语言之一。2011年1月，它被TIOBE编程语言排行榜评为2010年度语言，目前已经被广泛应用在系统管理工作、主程序开发的脚本语言、制作影片特效、管理讨论组、网络爬虫和搜索引擎的组件、计算机游戏和生物信息等各种领域。

之所以受欢迎是因为python具有很多优点：

1、面向对象的特性

面向对象是现在主流的程序设计思想，python语言在其设计过程中也使用了多态、封装、继承、类、接口等等这些面向对象的元素。

2、简单性

Python语句使用缩进的方式分隔代码块，没有像C语言一样使用大括号和分号，因而代码简洁短小，方便使用者阅读。

3、内置的数据结构

Python中的一些内置数据结构类似于java中集合类的功能，这些内置的数据结构包括元组、列表、字典等。元组相当于“只读”的数组，列表是python中常用的苦力，它可以作为可变长度的数组使用，处理多种数据，字典相当于java中的hashtable类型。

4、健壮性

能捕获程序的异常情况，提供异常退出机制。

5、跨平台性

Python解释执行的方式和java类似，在操作系统上首先会编译成与平台相关的二进制码，因此移植到其他平台上时也能正常运行。

6、动态性

python不需要另外声明变量，直接赋值即可以创建一个新的变量，而赋值时表达式的内容则决定了变量的数据类型。

7、应用广泛

python有许多第三方的库支持。例如：PIL库应用于图像处理，numpy库应用于数学计算，wxpython库应用于GUI程序的设计，Django库应用于web程序的开发等。

但python也有它的局限性和缺点：

1、单行语句问题

很多时候不能将程序连着写在一行里，尤其是使用模块导入语句和命令行打印语句时。

2、独特的缩进式语法

Python程序中严格使用缩进来区分代码块，最常见的情况是tab和空格的混用会导致错误，而这是用肉眼无法分别的，即便是很有经验的Python程序员，也可能陷入陷阱当中。

3、运行速度慢

由于python程序运行时要进行二次编译，编译器往往先将其解释为操作系统可执行的C语言程序，因此与C和C++相比起来，python程序在运行时速度要显然慢一些。对本系统而言，本科学生毕业论文的篇幅一般都不会过长，因此系统在扫描分析的过程中用时不会过长，还在使用者具有良好体验时间的范围内。

Python还具有很多功能强大的实用的内置标准库，本文简单介绍一下本系统在实现过程中必要的几个module。

1、zipfile

zipfile是python里用来做zip格式编码的压缩和解压缩的模块，其中有两个非常重要的类：ZipFile和ZipInfo。ZipFile是主要的类，用来创建和读取zip文件，而ZipInfo是存储的zip文件的每个文件的信息。由于word文档本身是一个压缩包，系统在分析时首先要将其解压缩，然后才能读取到其中的xml文件，因此使用到了该模块。

2、re

re模块包含对正则表达式的支持。就其本质而言，正则表达式是一种小型的、高度专业化的编程语言，使用这个正则表达式（或re模块）可以为想要匹配的相应字符串集指定规则，并对指定文本匹配符合该模式的字符串或对已经匹配到的文本做修改和分割等处理。本系统在对论文内容进行定位的过程中使用了正则表达式。

3、sys

使用sys（标准库模块）能够访问与python解释器和命令行相关的变量和参数。由于系统最终要封装成可执行程序，通过命令行提供入口参数的方式来调用执行，因此使用到了该模块。

## 2.2 lxml类库

作为第三方非标准库，使用由python编写的lxml可以实现对XML文件和数据结构迅速而又灵活地处理分析。本系统在读取分析word文档的xml数据结构时使用到了该类库的一些函数方法，因此这里简单介绍一下lxml的相关技术。

作为一款表现出高性能特征的 Python XML 库，lxml天生支持 XPath 1.0、XSLT 1.0、定制元素类，以及 python 风格的数据绑定接口。它构建在两个 C 库之上：libxml2 和 libxslt。它们为执行解析、序列化和转换等核心任务提供了主要动力。

使用lxml有很多常用的方法，本系统在实现过程中涉及到的函数方法主要有：

1）通常使用lxml.etree的方式对XML数据结构进行迭代

2）使用主要容器Element类来实现对大部分的XML tree的访问功能

3）使用tostring()方法来查看得到的XML

4）使用insert()方法插入新的子节点

5）XML元素支持属性，使用Element工厂方法可以直接创建。

6）使用set和get方法访问这些属性

7）使用attrib性质的字典接口

## 2.3 word文档的xml结构标签定义

Microsoft word文档使用HTML语言对其中编辑的文本信息进行储存，并保存在相应的xml文件中。本系统在读取到xml文件进行分析时，必须首先清楚其节点规则的定义，然后才能做相应的处理，进行下一步提取到有效的信息。官方对节点标签定义的解释说明现在可以公开的网站上进行阅读和查询，但并没有具体的说明文档，而在网站上的信息列表主要侧重完整性，因此显得标签条目众多，对每条标签的解释说明也侧重形式化定义和规则约束的统一性完备性，因此查阅起来有一定难度。

笔者在阅读和分析word文档官方网站上对节点定义描述的基础上，通过具体的的word文档实例与其解压后xml文件结合分析，整理出在学生毕业设计的论文文档中，主要涉及的节点定义如表2.1所示。

**表2.1 xml节点标签定义**

|  |  |  |
| --- | --- | --- |
| 节点或属性名称 | 说明 | 注解 |
| <w:document> | 主文件 | 此部分为word文件的主体部分 |
| <w:body> | 主体 | 正文部分 |
| <w:p> | 段落 | 代表一个自然段落 |
| <w:pPr> | 段落格式  文本段落属性 | 包含段落属性相关节点的信息 |
| <w:pStyle> | 段落样式 | 内置样式的识别码 |
| <w:ind> | 缩进 | 代表非标题段落的缩进属性 |
| <w:r> | 样本 | 一个段落内连续的一小段文本 |
| <w:rPr> | 样本格式 | 一个段落内样本的格式 |
| <w:rFonts> | 字体 | 复杂的字体格式 |

**表2.1 xml节点标签定义（续）**

|  |  |  |
| --- | --- | --- |
| 节点或属性名称 | 说明 | 注解 |
| <w:t> | 字符串 | 文本内容 |
| Xml:space | 空格 | 代表段落中出现的空格 |
| <w:b> | 粗体 | 其值表示是否加粗 |
| <w:cs> | 字号 | 其值代表文本字号大小 |
| <w:jc> | 对齐方式 | 代表段落的对齐方式 |
| <w:bookmarkStart>  <w:bookmarkEnd> | 书签 | 目录与正文中标题的对应 |
| <w:hyperlink> | 超链接 | 目录中标题的引用 |
| <w: vertAlign > | 上下标 | 参考文献引用时标号使用上标 |
| <w:space> | 间距 | 包含行距、段前段后等信息 |
| <w:table> | 表格 | 表示表格对象的引用 |
| <w:object> | 对象 | 表示对插图等对象的引用 |

## 2.4 pyinstaller工具

Pyinstaller是一个专用的封装python脚本程序的工具，本系统最终要封装为可执行程序，提供给论文格式修改系统进行调用，使用pyinstaller工具可以实现对完整的python程序进行封装。封装后的可执行程序可以移植到其他操作系统和平台上直接解释执行，因此使用pyinstaller非常方便，本文简单介绍一下pyinstaller工具。其优点主要是：

1）pyinstaller 是跨平台的可以用在linux和windows系统上

2）操作非常简单，几个命令就搞定了，这个比py2exe容易用多了

缺点是：

1）打包后的体积过大，因为要带python.dll库，所以没有办法

2）不能和tkinter 等库，打包成单独一个文件

在命令行中将python程序文件封装为可执行程序时的通用操作方法为：

pyinstaller.py [-opt] Pythonfile.py

其中[-opt]为附加参数，常用的参数命令解释如表3.2所示：

**表3.2 pyinstaller参数命令**

|  |  |
| --- | --- |
| 参数命令 | 说明 |
| -F | --onefile Py代码只有一个文件 |
| -D | --onedir Py代码放在一个目录中 |
| -K | --tk 包含TCL/TK |
| -X | --upx 使用upx压缩exe文件 |
| -v | --version=FILE 加入版本信息文件 |
| -d | --debug 生成debug模式的exe文件 |
| -w | --windowed, --noconsole 窗体exe文件 |
| -c | --nowindowed, --console 控制台exe文件 |

# 3 系统设计

本章将在前两章的基础上，对本系统设计时的主要思路进行描述，同时介绍系统主要模块所能够实现的功能，并且针对设计中的存在的关键技术问题描述解决方案。

## 3.1系统构架

通过分析目前在论文格式检查方面的相关研究，结合课题研究的三个意义，从可行性、后续开发性等角度考虑，决定设计一个基于python的论文格式检查系统，并采用前台与后台分离，前台通过服务调用后台python脚本来实现格式检查功能的结构。整体架构如图3.1所示。

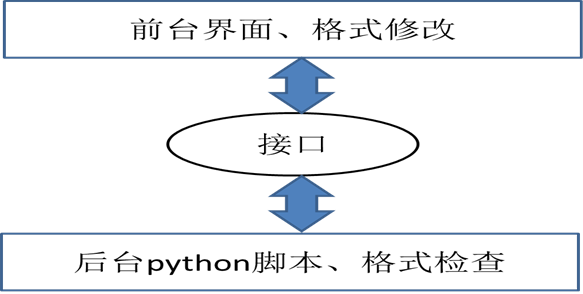


图3.1 系统构架

对于一篇待检查的学生毕业论文，首先由前台从服务器端接收提交的文档，同时生成规则接口，然后调用后台python程序执行，对该文档进行格式检查。检查程序以规则接口作为比较的衡量标准，对提交的论文分析文档内容，找出其中与规则模版不相匹配的部分，然后把检查的结果生成报告，并以接口文件的形式返回给前台。前台的格式修改系统通过读取检查报告，对论文内容的格式做进一步修改处理。

采用这种分离且相互独立的系统架构，易于系统功能的实现，同时对于维护工作和后续的程序更新也非常有利。由于工作量大，因此本文负责的是论文格式检查部分，并单独成为一个功能完善的系统。

## 3.2 系统模块

论文格式检查系统由于格式规则的制定改变需要保证系统设计具有可扩展性、良好的后续开发性，从性能的角度需要保证对文件的处理过程尽可能高效、准确。经过分析，系统的功能模块图如图3.2所示。

毕业论文格式检查系统

定制文档规则

格式提取比对

检查结果反馈

读读取待查文档

记记录错误信息

生生成报告

反反馈给前台

定定位论文内容

读读取文本格式

格格式匹配

解解压缩

读读xml内容

读读取标准格式

图3.2系统功能模块图

从系统模块图来看本系统主要分为定制文档的规则、文本格式提取和比对、检查结果反馈这三大部分，下面对这几部分作详细说明。

### 3.2.1 定制文档规则

根据前期工作中对于毕业论文格式的研究和规范化管理等方面的要求，需要在论文封面、声明、摘要、声明、目录、正文等几个部分实现毕业论文格式的检查。

论文封面格式和内容的检查，主要是对于论文封面中学生编辑的毕业设计字样、毕业设计论文题目、个人信息、时间日期等这几项内容进行字体、字号、对齐方式方面的比对和格式检查，。

摘要和声明部分首先要检查论文标题格式的是否符合规范，其次是声明和摘要正文的段落属性是否跟模版相匹配，对于摘要还要扫描关键字信息部分，查看关键字的编辑是否符合要求。

目录要根据预先设定好的格式，检查标题级数是否符合规范，检查目录是否为自动生成，即目录的页面与论文正文实际的标题能否对应。

正文中首先要检查各级标题的字体、字号、段前段后间距等段落属性的格式是否和模版匹配，检查各级标题是否使用了标题样式以及使用的样式级别是否正确，检查标题的标号开头是否符合要求，标号与标题内容直接是否有空格。然后是检查正文段落的中英文字体样式、字号大小、对齐方式和行间距、首行缩进等段落属性是否规范。同时还要检查正文中出现的图表标题格式是否符合规范，图标标题有无使用脚注，即是否使用了域来控制标号的自动生成，以及标题的序号是否和章节标题一致。

还有参考文献部分，检查每一条参考文献条目的字体、字号等属性是否符合要求，检查是否用正确的标号格式开头。

规则制定的过程是与前台一起协商的，由于系统主要是作为后台脚本调用，在程序运行时，由前台界面动态化的生成一个规则模版，并创建相关的接口路径和配置文件。脚本程序读取并分析其中的每一条规则，生成模版规则字典，用来在格式比对的时候进行匹配。主要检查的项目和内容如表3.1所示。

**表3.1 检查项目**

|  |  |
| --- | --- |
| 项目 | 内容 |
| 封面毕业设计 | 检查其字体、字号、对齐方式 |
| 封面论文标题 | 检查其字体、字号、对齐方式 |
| 封面款注 | 检查院系名称、专业名称、学生姓名、  指导教师等字体、字号、对齐方式 |
| 封面书脊 | 检查字体类型、字体大小、对齐方式 |
| 本人声明 | 检查字体类型、字体大小、是否加粗、  是否居中 |
| 声明正文 | 检查字体、字号、对齐方式，  以及锻炼的行距、首行缩进 |
| 中文摘要 | 检查标题的字体、字号、对齐方式  检查正文的字体、字号、对齐方式 |

的**表3.1 检查项目（续）**

|  |  |
| --- | --- |
| 项目 | 内容 |
| 关键字 | 检查字体、字号、对齐方式、是否加粗  检查关键字内容的字体、字号 |
| 英文摘要 | 检查标题的字体、字号、对齐方式  检查正文的字体、字号、对齐方式  以及段落的行距、缩进 |
| 英文关键字 | 检查字体、字号、对齐方式、是否加粗  检查关键字内容的字体、字号 |
| 目录 | 检查标题级数、字体、字号  检查目录是否为自动生成 |
| 章节标题 | 检查检查字体、字号、对齐方式、是否加粗  检查段前段后间距  检查标题开始格式 |
| 正文 | 检查中文字体、字号、对齐方式  检查英文字体类型、字体大小  检查段落的属性 |
| 图表 | 检查图表标题的格式、字体、字号 |

### 3.2.2 文本提取和格式比对

Word文档本质上是一个压缩包，解压后的目录如图3.3所示：

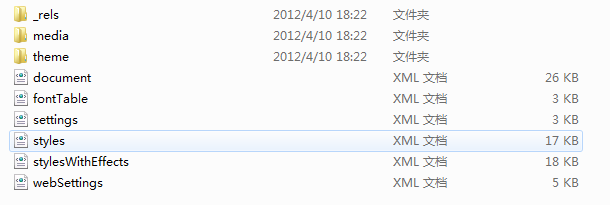


图3.3 word目录结构

其中document.xml记录了test.docx的主要文档内容，styles.xml存储标准标签（如title，heading1，heading2等等）和文档段落、文字和图片等元素的属性；setting.xml记录与应用程序相关的设置，如兼容性模式，字体特性等；fontTable.xml记录与应用程序相关的字体集合；styleWithEffect.xml与styles.xml记录内容相近，推测是与ms office 2010相关的文档显示特性；webSettings.xml记录内容含义未查明。

使用python内置的标准库zipfile完成对word文档的解压缩工作，解压后的文件夹即包含document.xml和styles.xml等文件。然后使用组件lxml类库来完成读取其中word目录下我们主要分析的文件document.xml和styles.xml的里面的内容。

在读取XML内容的同时，使用lxml内建函数完成对xml的树形数据结构进行迭代分析的工作，根据标签判断和检索符合需求的节点，并使用python的yield生成器从迭代序列中输出该节点。

在对xml数据结构进行迭代时，遵循一遍扫描的原则，因此需要在每分析一个<w:p>节点的过程中，需要完成以下三项工作：

1、提取出该段落节点的文本信息

系统是以段落文本为单位进行分析的，不管是接下来的内容定位工作，还是检查结果出来后给使用者进行直观的反馈，还有在系统测试时寻找问题所在，都需要提起出该段落的文本信息。

2、从节点中提取需要的格式信息

格式检查是系统完成的主要工作和主要目标，在分析一个段落节点时必然要能够完成对其格式信息提取的工作。与格式相关的信息主要是该段落文本的字体样式、字号大小、对齐方式、字形是否加粗等字体属性已经行间距、段前段后间距、首行缩进等段落属性。

3、分析文本特点定位段落位置

毕业论文分为若干部分，每一部分有不同的段落类型，段落之间的格式信息有所差异，为了实现格式匹配检查的目的，系统必须要在扫描过程完成段落定位的工作，根据该段落文本的内容及其特征，结合每一个段落类型所独有的特性，识别出该节点属于毕业论文中哪一部分的段落。

论文内容定位和格式检查的过程如图3.4所示。

提取格式信息

定位文本位置

解析XML节点

读取规则配置

规则模版

图3.4 格式比对过程

### 3.2.3 检查结果反馈

系统对于错误的格式记录其位置、错误类型以及列出该错误应该改成什么样的标准格式，并以接口文件的形式反馈给前台再进行后续的处理工作。对于接口文件的定义，需要与前台进行协商，包括文件应记录的内容、记录的格式以及数据储存的结构。

当系统在运行时当获取了一个<w:p>节点包含的文本、格式信息，然后又找到了该节点在论文内容中的位置后，就根据模版中的规则来进行格式检查分析的工作。在检查比对过程中使用错误信息列表来收集发现的问题，最终把收集到的这些问题按照约定好的数据格式写入生成报告的文件中，后续处理工作分析该报告文件，对出现错误问题的指定位置进行格式修改。同时为了方便使用者查看问题所在，系统把收集到的信息也以较为直观方便查看阅读的形式写入另一个报告文件中，反馈给写作该论文的学生用户。

## 3.3 关键技术解决方案

1. 整理word文档的xml标签定义：

Microsoft word文档本质上是一个压缩包，并默认以ZIP格式进行压缩编码的，为了能够读取其中的xml数据结构，需要首先能够把待分析的word解压缩并根据路径选择相应的xml文件进行分析。将一个word文档解压缩后，其中的docement.xml文件包含了文档中的文本及其相关信息，是我们主要分析的对象。而docement.xml中以XML数据结构储存信息，类似于HTML语言，要提取所需信息，就要知道word文档是如何定义这些节点的含义的。在官方网站上罗列出的规则条目很多，对每一个规则从全面的角度去总结其功能与含义，这样相对每个条目的解释来说就会比较空泛，较难弄清楚其具体含义，因此工作量和难度较高。

在系统项目设计实现的过程中，为了解决这一问题，采取了阅读分析官方网站标签解释和实际情况相结合的方式，来整理在系统分析中时需要理解其定义的标签。比如在正文中出现图表标题时，需要检查该处是否使用了脚注和域来控制图表标题的序号，这其中涉及到使用脚注在xml结构中的标签定义，通过创建一个word文档在其中编辑好使用脚注的文本，然后打开其xml文件寻找和分析相应的节点标签的方法，最终确定了使用脚注相关的标签定义，为系统在检查和判断时提供依据。

1. 从节点中提取文本对应的格式：

获取一个节点包含的格式信息比较复杂，也是一个技术难点。由于Microsoft word也没有在官方文档中说明与文本有关的字体属性以及段落有关的格式属性具体是如何在xml中组织和体现的，因此需要收集各种不同的情况作实际分析。在实际的测试过程中，发现主要存在以下几种情况：

1）在word中对编辑好的段落文本进行各种显示的格式修改，那么相关的格式信息都会体现在该<w:p>节点包含的<w:pPr>节点中。

2）在word中对一段文本直接使用预设样式，相应的格式信息在styles.xml文件中。

3）直接打开word编辑一段文字，格式为默认。

根据这些情况，系统实现时的解决方案如图3.5所示。

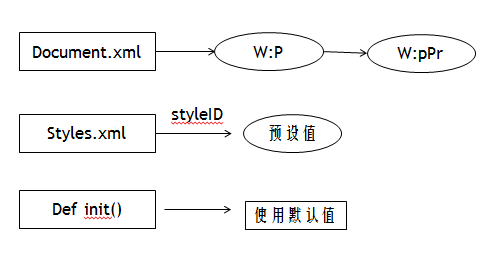


图3.5 格式提取解决方案

1. 识别和定位论文内容：

如果把程序分析XML的过程和编译过程进行比较的话，会发现有许多相似的地方，然而编译是有文法基础支持的，在文法分析的过程中有规则可循，机器可以很容易实现对文法的规则检查。但在识别论文的时候没有一个理论上严谨的无二义性文法，只能通过人为设定的一些规则或手段去匹配论文的内容，识别出该部分文本是论文中的哪一部分，从而再对其进行格式的检查与判断，这在研究过程中也是一大技术难点。

解决这一问题的方法是采用把毕业论文分块检查和分段落类型的方法来定位文本内容。毕业论文的内容大致可以分为封面书脊、声明、中文摘要、英文摘要、目录、正文几部分。在正文之前的所有内容都是限定好的，学生在写作论文的时候很少出现在该部分内容出错的情况，因此使用一个模版来定位论文内容，模版中对每一部分所包含的详细信息进行标记跟踪，读取到相应的内容后给标记赋值，同时检测是否读到下一部分的内容。在目录之后的正文部分内容比较多但段落类型确定，主要是标题、正文段落、图、表、图标题、表标题这几种段落类型，因此通过使用正则表达式来匹配每一个段落类型独有的特征值的方法实现正文部分段落类型的划分，同时参考相应的<w:jc>、<w:b>、<w:ind>、<w:space>等标签属性的值来提高定位准确性。

# 4 系统实现

本系统的实现主要涉及读取xml、读取文本、获得格式、文本内容定位、读取规则、检查生成报告等模块，本章将依次介绍这些模块实现的方法，实现相应功能时所定义的函数和数据结构。

## 4.1 系统整体运行过程

根据第三章设计的系统模块图，结合实际情况，系统在实际运行的过程中，首先读取从前台生成的规则接口rules.txt，生成一个格式字典rules\_dct。然后系统调用函数解析word文档，从中读取到word文档包含的document.xml文件和styles.xml文件，并对document.xml文件以<w:p>节点为最小单位进行迭代分析。在迭代分析<w:p>节点的过程中，调用获取文本、获取格式、定位内容、检查比对等函数对每一个<w:p>节点进行了具体分析，同时为了满足广义上检查格式的需求，针对内容位置的不同情况又作了相应的不同分析，比如当段落文本属于关键词时，深入到<w:p>节点内部检查关键词内容的格式；比如当段落文本属于图表标题时，检查是否使用了脚注以及标题序号是否与章节标题一致。最后使用report列表将检查过程的warning信息和error信息都收集起来，输出结果分为两部分：一是按定义好的JS数据格式组织存储在生成报告的文档中，以便后续格式修改系统的分析处理；二是将检查过程中的错误报告信息以直观的形式写入txt文档中，方便使用者查看。

## 4.2 读取xml模块

Microsoft word文档本质上是一个压缩包，并默认以ZIP格式进行压缩编码的，为了能够读取其中的xml数据结构，程序首先能够把待分析的word解压缩并根据路径选择相应的xml文件进行分析。

使用python自带的zipfile类库可以完成解压缩的过程，程序以调用函数的方式将word文档以解压缩后的编码方式打开，并根据路径读取相应的xml文件，其中word\document.xml所包含的信息是word文档里的所有文本和相关格式信息，而word\styles.xml文件包含有word文档本身内设的一些样式的格式信息，程序调用函数将读取到的xml文件以字符串的方式返回。

上一步返回的xml文件数据还是字符串格式的，而xml数据结构本身是一种树形结构，信息都包含在节点内容以及节点的属性里，节点之间以树形结构穿联起来，因此为了获取具体的节点进行分析，程序首先借助了lxml模块的etree函数还原xml的树形结构，然后通过节点的标签来对xml进行迭代分析，当截取到所需节点时，使用python语言的yield生成器在iter序列中输出该节点，进行下一步具体分析。

这一模块包括的函数以及对应代码如下：

|  |
| --- |
| def get\_word\_xml(docx\_filename):  zipF = zipfile.ZipFile(docx\_filename)  xml\_content = zipF.read('word/document.xml')  style\_content=zipF.read('word/styles.xml')  return xml\_content,style\_content  def get\_xml\_tree(xml\_string):  return etree.fromstring(xml\_string)  def \_iter(my\_tree,type\_char):  for node in my\_tree.iter(tag=etree.Element):  if \_check\_element\_is(node,type\_char):  yield node  def \_check\_element\_is(element,type\_char):  return element.ta== '%s%s' %(word\_schema,type\_char) |

## 4.3 获取节点包含的文本及格式信息

每个<w:p>节点表示word文档中相邻两个回车换行符之间的全部内容，即一个自然段落，在xml中由一个<w:pPr>节点和若干个<w:r>节点组成。而一个<w:r>节点又包含一个<w:rPr>和一个<w:t>节点，<w:r>相当于把<w:p>分割成若干个具有连续特点的样式文本，所以每个<w:r>节点包含一个<w:t>节点，其中就是word中的文本信息了，遍历一个<w:p>节点中的所有<w:t>节点，把所有字符串连接起来，就获得了word中一个自然段的全部文字。该函数的代码如下，入口参数为一个<w:p>或<w:t>节点，返回该节点包含的全部文本。

|  |
| --- |
| def get\_ptext(w\_p):  ptext = ''  for node in \_iter(w\_p,'t'):  ptext += node.text  return ptext.encode(Unicode\_bt,'ignore') |

获取一个节点包含的格式信息比较复杂，也是一个技术难点。由于Microsoft word也没有在官方文档中说明与文本有关的字体属性以及段落有关的格式属性具体是如何在xml中组织和体现的，因此需要收集各种不同的情况作实际分析。在实际的测试过程中，发现主要存在以下几种情况：

1）在word中对编辑好的段落文本进行各种显示的格式修改，那么相关的格式信息都会体现在该<w:p>节点包含的<w:pPr>节点中，如图4.1所示。

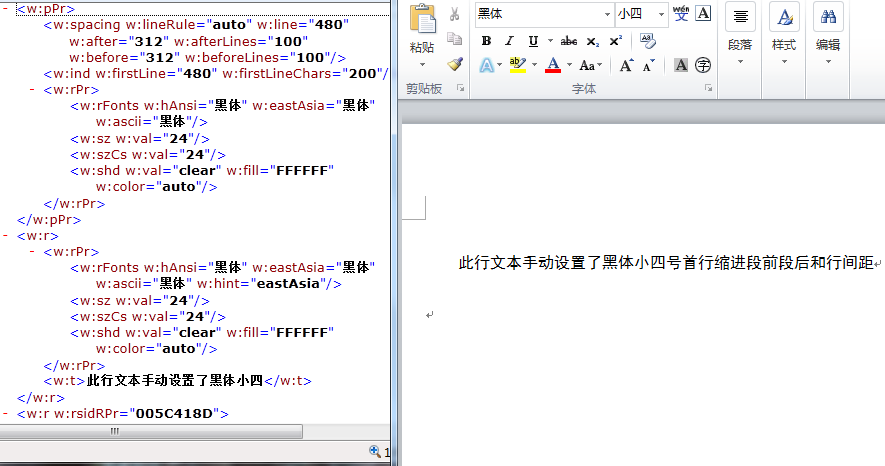


图4.1 显示修改格式

2）在word中对一段文本直接使用预设的各种样式，那么在docement.xml中会出现一个<w:pStyle>节点，需要去styles.xml文件中根据<w:pStyle>的ID去寻找对应的预设样式包含的格式属性标签，如图4.2所示。

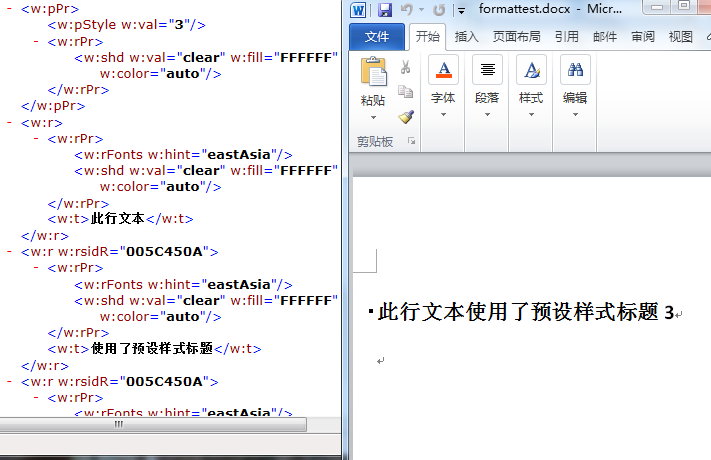


图4.2 使用预设样式

特别的，在styles.xml中各ID对应的预设样式之间存在互相关联、一种样式基于另一种样式的情况，在根据<w:pStyle>的ID值寻找预设样式的同时，还要根据该预设样式中包含的<w:basedOn>信息继续寻找相关的格式属性。

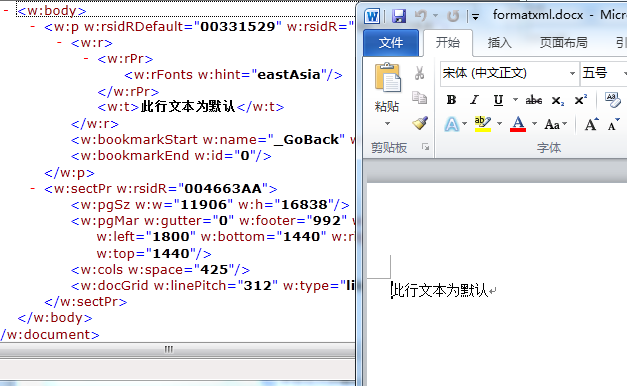


图4.3 默认文本格式

特别的，如果一段文本即使用了预设样式，又在编辑的过程中对字体、段落属性等格式信息作了显示修改，那么修改过的部分会出现在该<w:p>节点包含的<w:pPr>节点中。系统在运行过程中充分考虑到这一点，以document.xml作为获取格式的优先级更高。

3）如果直接打开word编辑一段文字，既没有使用预设样式也没有对字体、段落属性等格式信息作任何显示修改，那么暂时没有发现相关的xml文件有这种情况对应的格式信息，如图4.3所示。系统在运行过程中使用初始化来解决这一问题。

结合以上各种情况，系统在获取格式时，首先把<w:p>节点作为参数传入函数get\_format()中，在函数开头调用一个初始化函数init\_fd（）把格式字典进行初始化赋值，即对格式字典内字体、字号、对齐方式、加粗、段间距、段前、段后、首行缩进、段落级别这些键赋一个默认值，默认值为打开word文档直接编辑一段文本时的默认格式。由于未能在xml文件中分析出默认格式每一个字段具体对应的值，因此在赋初值的时候，采用目前Microsoft word官方文档在大多数机器上的docx格式的文档的默认编辑文本的格式，其中中文字体初值设为宋体，字号为小四，对齐方式为两端对齐，行间距为单倍行距，段前间距零字符，段后间距零字符，段落级别为正文。（注：为方便程序处理值的类型均为字符串类型）初始化的赋值说明如表4.1所示。

**表4.1 格式字典赋初值**

|  |  |  |
| --- | --- | --- |
| 字典键 | 初值 | 说明 |
| fontCN | 宋体 | 默认中文字符为宋体 |
| fontEN | Times New Roman | 默认英文字符字体 |
| fontSize | 21 | 默认字号 |
| paraAlign | both | 默认对齐方式为两端对齐 |
| fontShape | 0 | 默认字体不加粗 |
| paraSpace | 240 | 段间距为单倍行距 |
| paraIsIntent | 0 | 首行缩进0字符 |
| paraFrontSpace | 100 | 段前间距0字符 |
| paraAfterSpace | 100 | 段后间距0字符 |
| paraGrade | 0 | 段落级别为正文 |

对格式字典进行初始化后，系统程序开始分析作为参数传入的<w:p>节点，由于在xml结构定义中一个<w:p>节点只包含一个<w:pPr>节点，因此调用iter函数找出表示该段落文本格式信息的<w:pPr>节点，对其进行深度分析。

在分析的过程中优先查找是否包含<w:pstyle>节点，即该段落文本是否使用了预设样式，如果查找到包含有<w:pstyle>节点，那么系统记录该节点表示的ID值，然后转入到styles.xml文件中，寻找ID值对应相同的预设样式。在styles.xml中每个预设样式都有一个固定ID值，预设样式的格式同样也是包含在<w:pPr>节点中的，因此找到对应的ID值后，调用assign\_fd（）函数对格式字典进行赋值，修改过的字段值会覆盖原先的初始值。

如果一个段落未使用预设样式，系统的分析的过程中就不会找到<w:pstyle>节点，也就不会转入到styles.xml文件中寻找格式，而是直接调用assign\_fd（）函数从该段落文本自身包含的格式属性<w:pPr>节点中寻找每一个字段修改后的值，并覆盖到格式字典中去。

assign\_fd( )函数在调用时，系统把<w:pPr>节点作为参数传入，遍历该节点的每一个子节点，根据第二章对xml标签的定义描述，字体、字号、对齐方式等格式信息都作为子节点组织在<w:pPr>节点下，遍历时调用\_check\_element\_is()函数，判断子节点是否属于符合查找要求，如果符合则取出其中对应的值，不符合则继续判断下一个子节点。而针对不同的格式节点，取值的过程也略有不同，对于字号节点<w:sz>、对齐方式节点<w:jc>、段落级别节点<w:outlineLvl>可以直接通过特征名val取出其中的值赋给格式字典对应的字段；对于字形节点<w:b>，如果其值为on，则把fontShape字段赋值为1，表示该段文本加粗；对于缩进节点<w:ind>查找其是否包含<w:firstlineChars>，如果包含则直接把paraIsIntent字段赋值为1，表示该段落文本首行缩进；对于段落间距节点<w:spacing>，由于间距的表示有磅值、厘米等不同单位，因此在xml标签定义中有多个特征名来表示，系统在分析时取磅值作单位，给paraSpace等字段进行赋值；而对于字体节点<w:rFonts>的分析就显得尤为复杂了，这是由于本身xml标签在定义中考虑到了多种语言的不同字体表示，系统在分析时为了简化运行过程，对中文字符先查找特征名eastAsia进行赋值，若不包含则查找特征名hint把fontCN赋值为宋体，对英文字符查找特征名hAnsi赋值给fontEN字段。

该模块的主要函数代码如下：

|  |
| --- |
| def get format(node,d):  init\_fd(d)  for pPr in \_iter(node,'pPr'):  for pstyle in \_iter(pPr,'pStyle'):  styleID = get\_val(pstyle,'val')  found=0  for style in \_iter(style\_tree,'style'):  if get\_val(style,'styleId') == styleID:  found=1  for nod in \_iter(style,'pPr'):  assign\_fd(nod,d)  for nod in \_iter(style,'rPr'):  assign\_fd(nod,d)  if not found:  print 'No such a styleId in styles.xml!'  assign\_fd(pPr,d)  return d |

从该函数的运行过程可以看出，在获取一段文本的格式信息时，首先把初始值赋值给格式字典，然后用预设样式的格式给格式字典赋值，最后再用节点本身的格式信息去给格式字典赋值，后一次的赋值是在前一次的格式字典上采用覆盖的方式，用这样的方法就保证了在上文中列举的几种情况下能够正确获取一段文本的格式。比如一段文本直接编辑保存，没有显示设置相关字体、段落属性的格式，那么在系统分析时，由于首先赋了初值，然后并没有找到<w:pstyle>节点的预设样式，也没有在节点本身的<w:pPr>中获取到任何信息，此时格式字典中字段的值还是原来初始化时赋给的值，正好符合上述第三种情况的要求，分析结果是正确的。

## 4.4 定位论文内容

在读取到一段文本并获取到它的格式信息后，需要对该段在论文内容中的位置进行判断，从而取出规则模版中该位置的正确格式与之进行比较，把一部分的功能称为文本定位。

为了更准确的分析出一段文本属于论文内容中哪一部分，提高定位的准确率，首先对论文作一个大的划分，把一篇毕业论文分为封面、声明、摘要、目录、正文、参考文献这几部分，然后对每一部分内部划分出小的段落类型，比如封面部分包含论文标题、毕业字样、作者个人信息、时间日期、封面书脊这几个段落类型，摘要部分包含论文标题、摘要标题、作者信息、摘要正文、时间日期这几个段落类型，正文部分则包含一级标题、二级标题、三级标题、图标题、表标题、正文段落这几个段落类型。对论文进行大part划分和小段落类型指定后，文本分析定位时判断的区域大大缩小，每一个区域内文本段落的独有特点都比较明显，从而提高判定条件的独立性，从而大大提高了定位的准确性。对论文内容的划分如表4.2所示。

**表4.2 论文内容划分**

|  |  |
| --- | --- |
| 节 | 包含的段落类型 |
| 封面 | 封面\_单位代码  封面\_毕业设计  封面\_论文标题  封面\_个人信息  封面\_日期  封面\_书脊 |
| 声明 | 个人声明\_标题  个人声明\_正文  个人声明\_签名 |
| 摘要 | 摘要\_论文题目  摘要\_学生及教师姓名  摘要\_标题  摘要\_内容  摘要\_关键词  摘要\_关键词内容 |
| 目录 | 目录\_标题  目录\_一级标题  目录\_二级标题  目录\_三级标题 |
| 正文 | 正文\_一级标题  正文\_二级标题  正文\_三级标题  正文\_段落  附录\_内容  图标题  表标题 |
| 参考文献 | 正文\_一级标题  参考文献\_条目 |

在定位的方法上，又根据目录前和目录后内容性质上的不同，使用了不同的方法。在目录之前的内容，基本上都是固定的、统一的内容，比如封面上必然包含右上角的个人信息、毕业设计字样、论文标题、中下的个人信息、日期这几部分内容，各部分之间划分比较明显，学生在写作的过程中，在这些地方也很难出现内容上的错误，因此对目录之前的几个部分，采用模版顺序检测的定位方式，即某一部分包含某一项内容，那么接下来必然是另一项已知的定义好的内容。

而在目录之后的正文部分，主要由一级标题、二级标题、三级标题、图表标题、图表对象、正文段落这几项组合而成，根据每一项所特有的区别于其他项的特征值，采用正则表达式匹配的方法来判别各级标题，不能匹配为标题的段落则划分为正文段落。使用正则表达式匹配和分析的函数代码如下：

|  |
| --- |
| def analyse(text):  text=text.strip(' ')  if text.isdigit():return 'body'  pat1 = re.compile('[0-9]+')  pat2 = re.compile('[0-9]+\\.[0-9]')  pat3 = re.compile('[0-9]+\\.[0-9][\\.[0-9](file:///\\.[0-9)]')  pat4 = re.compile('图(\s)\*[0-9]+(\\.|-)[0-9]')  pat5 = re.compile('表(\s)\*[0-9]+([\\.|-)[0-9](file:///\\.|-)[0-9)]')  if pat1.match(text):  if pat1.sub('',text)[0] == ' ': sort = 'firstLv'  elif pat1.sub('',text)[0] =='.':  if pat2.match(text):  if pat2.sub('',text)[0] == ' ': sort = 'secondLv'  elif pat2.sub('',text)[0]=='.':  if pat3.match(text):  if pat3.sub('',text)[0]==' ': sort = 'thirdLv'  elif pat3.sub('',text)[0]=='.': sort = 'overflow'  else: sort ='thirdLv\_e'  else: sort='secondLv\_e'  else: sort = 'secondLv\_e'  else: sort = 'body'  else: sort = 'firstLv\_e'  elif pat4.match(text): sort = 'objectT'  elif pat5.match(text): sort = 'tableT'  elif text=='参考文献' : sort = 'refer'  else : sort ='body'  return sort |

另外由于参考文献的格式与正文段落不同，需要进行检查的格式和规则也有所变化，因此把参考文献作为单独的一个part划分出来，参考文献中包含标题和每一条参考条目，标题的段落类型划分为正文\_一级标题。系统在分析时检查每一条参考文献的编辑格式是否正确，同时使用正则表达式检查参考文献是否使用中括号的标号开头。在正文部分和参考文献部分用来匹配段落类型的正则表达式如表4.3所示。

**表4.3 匹配时所使用的正则表达式**

|  |  |  |
| --- | --- | --- |
| 字段名 | 段落类型 | 匹配采用的正则表达式 |
| firstTitle | 正文\_一级标题 | [0-9]+ |
| secondTitle | 正文\_二级标题 | [0-9]+\\.[0-9] |
| thirdTitle | 正文\_三级标题 | [0-9]+\\.[0-9]\\.[0-9] |
| objectTitle | 图标题 | 图(\s)\*[0-9]+(\\.|-)[0-9] |
| tableTitle | 表标题 | 表(\s)\*[0-9]+(\\.|-)[0-9] |
| reference | 参考文献\_条目 | \\[[0-9]+\\] |

使用正则表达式有以下几个优点：

一是提高匹配字符串的效率

二是以后段落类型划分的规则若有修改的话，直接修改用来匹配的正则表达式即可，不需要去大段修改代码，提高了系统的可维护性。

三是易于未来的拓展工作。目前本系统只适用于本科学生毕业论文的格式检查，并且在应用层面上还受到各学院之间对论文格式要求存在差异的限制。未来如果需要把系统进一步推广到更多的应用区间，或者推广到一般word文档的格式检查，使用正则表达式具有良好的集成性。

## 4.5 检查并生成报告

系统在运行时当获取了一个<w:p>节点包含的文本、格式信息，然后又找到了该节点在论文内容中的位置后，就调用check\_out函数，把规则字典、待检查的段落的格式字典、分析出的该段落在论文内容中的位置作为参数传入，根据规则字典中该位置处定义的好的格式来和待检查段落中的字体、段落属性等格式信息进行比对，查看是否符合要求。在函数开头定义了一个空的errorInfo信息收集列表，如果在检查比对的过程中发现存在不一致的问题，就把出现错误的字段和在规则模版中该字段应有的正确值收集到信息列表中去，最后函数把收集到的错误信息列表返回给主程序。该模块函数的代码如下：

|  |
| --- |
| def check\_out(rule,to\_check,locate):  errorInfo=[]  if locate in checkItemDct.keys():  for key in checkItemDct[locate]:  if key == 'paraAlign': print ' ',to\_check[key]  if to\_check[key] != rule[key]: print：key,to\_check[key]  errorInfo.append('\'type\':\''+errorTypeName[key]+'\',\'correct\':\''+rule[key]+'\'')  else: for key in errorTypeName:  if to\_check[key] != rule[key]: print key,to\_check[key]  errorInfo.append('\'type\':\''+errorTypeName[key]+'\',\'correct\':\''+rule[key]+'\'')  return errorInfo |

主程序在收到返回errorInfo信息列表后，首先判断列表是否为空，如果为空则说明该文本段落格式设置正确，不作任何处理（测试程序会在命令行打印报告出检查结果为正确），如果不为空，则说明存在问题，系统会把其中的每一条记录按接口中定义好的格式写入报告文件中，报告文件的路径由前台程序在调用的时候通过命令行参数指定。

上述错误信息列表errorInfo表示该段落文本存在字体属性、段落属性等基础性的格式错误，具体体现为格式字典中定义的十个字段。而在实际的分析过程中，为了满足需求和实现实际应用时的功能，系统还需要对一些人为规定的广义格式错误进行分析和收集，比如关键词后面要有冒号，关键词之间用逗号分隔开，正文中不允许出现连续的空行，图表标题需要使用引用，正文中对参考文献的引用标号必须使用上标，章节标题的标号应符合要求，标号和标题内容之间应该用空格隔开等等这些问题，系统在运行的过程中对这些问题也能够进行分析，为了便于区分和后续处理，把这些问题使用了一个warnInfo信息列表收集起来，并且用同样的方式写入到报告文件中。

为了接口文件格式的一致性，两种错误问题采用相同的格式写在报告文件中，使用一个错误级别字段level进行区分，其值包含error和warning两种。对于error级别的记录，后续格式修改系统可以直接修改xml文件的数据结构来达到修改格式的目的，同时还可以在论文对该处高亮显示；对于warning级别的记录，后续处理工作仅仅是把警告信息显示在论文相应位置上，需要学生本人去手动进行修改。本系统在实际运行时能处理的错误类型如表4.4所示。

**表4.4 错误报告类型**

|  |  |  |  |
| --- | --- | --- | --- |
| Type | Level | 出现的位置 | 说明 |
| font | error | \*all | 字体不符 |
| fontsize | error | \*all | 字号不符 |
| fontshape | error | \*all | 字形加粗不符 |
| gradeAlign | error | \*all | 对齐方式不符 |
| gradeSpace | error | 正文\_段落，摘要\_正文，  声明\_正文 | 行间距不合适 |
| gradeFrontSpace | error | 正文各级标题，正文段落 | 段前间距不合适 |
| gradeAfterSpace | error | 正文各级标题，正文段落 | 段后间距不合适 |
| FLind | error | 正文\_段落，摘要\_正文，  声明\_正文 | 未使用首行缩进 |
| Warn1 | warning | 摘要\_关键词 | 关键词后面没有冒号 |
| Warn2 | warning | 正文\_图标题，正文\_表标题 | 图表标题未使用引用 |
| Warn3 | warning | 正文\_图标题，  正文\_表标题 | 图表标题的序号和章节序号不一致 |
| Warn4 | warning | 正文\_段落 | 正文中参考文献的引用未使用上标 |
| Warn5 | warning | 正文\_段落 | 不允许出现连续空行 |
| Warn6 | warning | 正文各级标题  目录各级标题 | 标题标号需要和标题之间用空格隔开 |
| Warn7 | warning | 参考文献\_参考条目 | 参考文献没有以[num]标号开头 |
| Warn8 | warning | 目录部分 | 目录未更新 |

## 4.6 其他模块

系统还对目录是否自动更新进行了检查，从目录部分扫描<w:hyperlink>标签节点并统计在一个列表里，然后在正文部分扫描<w:bookmarkStart>标签节点并统计在另一个列表里（<w:bookmarkStart>和<w:bookmarkEnd>一般是成对出现的），最后检查两个列表的出入来判断目录是否自动更新。相关代码如下：

|  |
| --- |
| hyperlinks = [ ]  bookmarks = [ ]  for node in \_iter(xml\_tree, 'hyperlink'):  temp=''  for hl in \_iter(node,'t'): temp += hl.text  hyperlinks.append(node.values()[0])  #print True,temp  for node in \_iter(xml\_tree, 'bookmarkStart'):  bookmarks.append(node.values()[1])  catalog\_ud= True  for i in hyperlinks:  if i not in bookmarks: catalog\_ud =False  if catalog\_ud: pass  print True,'目录正常更新'  else: pass  print False,'目录未更新' |

# 5 系统测试

本章将使用若干测试样例，对系统各部分功能的实现情况进行测试。对一个实用性系统而言，测试的目的主要是检测系统各部分模块的功能是否正常运行以及运行的结果是否符合需求。

## 5.1 测试环境

操作系统：Windows 7 SP1

软件平台：python2.7.9

检测文件：Microsoft word 2010 docx文件

接口文件：使用ANSI编码方式的txt文件

封装工具：pyinstaller-2.1

## 5.2 系统功能测试

本系统在最终作为调用程序时用pyinstaller封装为可执行程序，只把检查完后生成的报告写入一个txt文件，其中是按接口定义好的JS格式的数据。为了方便测试系统功能和显示结果，测试程序专门使用了print打印语句，把读取规则接口、待检测的word文档后，将系统运行时的中间的信息和错误报告都打印在命令行里，方便测试查看。

### 5.2.1 格式提取功能

测试目的：测试格式提取模块能否正常工作，获得的格式是否和word文档中文本编辑的格式相一致

测试方法：本节作为单独模块的功能测试，因此把格式获取相关的函数单独放在一个程序文件里，并把输出的结果打印出来。在测试的word文档中编辑文本，分别使用显示设置、使用样式、默认等方式编辑格式，然后运行模块程序，查看输出是否符合预期要求。

操作步骤：

1）将模块程序、测试论文文档放在同一目录下

2）在测试文档中编辑好不同格式的段落文本

3）运行模块程序

4）查看输出结果

预期结果：

对word文档中用不同方式编辑的文本段落应该正确分析出其对应的格式。比如第一行使用默认方式编辑的文本，字体应该显示为宋体，字号21，对齐方式显示both等，与初始化函数init\_fd（）中设置的值一致；第二行使用了标题样式的文本，应该显示字体黑体，字号，对齐方式等；第三行手动编辑了格式的文本，格式各字段都应该显示手动设置的值。

运行结果：如图5.1所示。

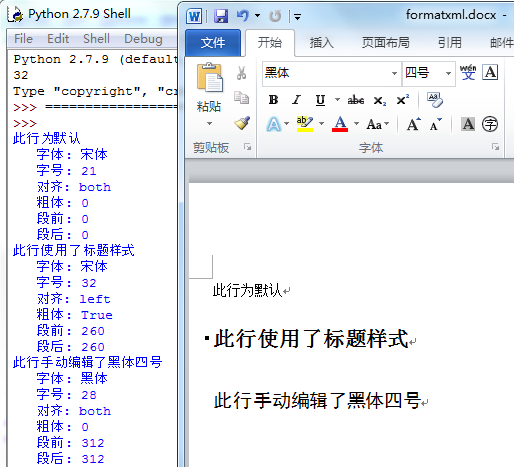


图5.1 测试结果一

结果说明：

运行的结果与预期一致，说明格式提取模块运行正常，功能正确，并能在不同情况都获取正确的格式信息，与系统需求一致。

### 5.2.2 定位论文内容功能

测试目的：测试系统在实际运行过程中，能否正确分析出待查论文中的段落在论文中相应位置，模块是否正常运行，分析出的结果是否正确，是否符合系统设计的需求

测试方法：本节也是单独的模块功能测试，因此把内容定位相关的函数单独放在一个测试程序里，使用一个具有完整内容的毕业论文样例，运行程序查看系统对位置的划分和定位是否正确。

操作步骤：

1）将模块程序、测试论文文档放在同一目录下

2）在测试文档中是一篇完整的论文样例

3）运行模块程序

4）查看输出结果

预期结果：

对一篇完整的并且内容上符合书写要求的论文，系统应该能正确分析出其中的封面、摘要、声明、目录、正文、参考文献部分，并对每一部分内部的段落类型比如正文中的一级标题、二级标题、三级标题、图表标题、正文段落等都能够正确识别，显示结果应和论文内容中的位置一致。

运行结果：

显示的部分结果如图5.2所示。

结果说明：

在输出的运行结果中，对测试论文样例的各部分以及每一部分内部的段落类型都能正确分析出相应的位置，该模块运行正常，功能正确，符合系统设计的要求。

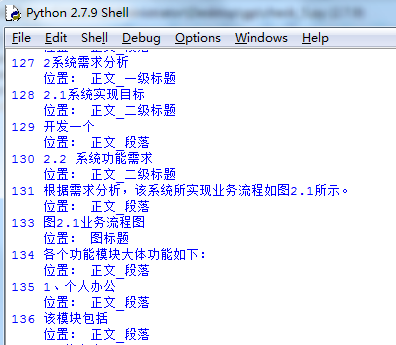


图5.2 测试结果二

## 5.3 系统性能测试

本节对系统在实际运行过程中的耗时进行测试。由于python语言本身的特性，在执行时需要进行二次编译，运行速度稍微慢于一般的C语言程序。另外由于毕业论文篇幅的影响，如果对于一篇较长篇幅的论文文档进行检测时，系统运行的用时大大增加，这对使用者来说的体验是非常不好的。再加上如果用时过久，系统响应速度过慢，后续的格式修改系统和处理程序可能要等待更长的时间，影响到分析的效率。因此对系统性能进行测试，保障在实际运行时的体验效果良好。

测试目的：对系统实际运行的用时进行测试，保障系统性能在可接受的范围内

测试方法：在系统中添加计时的代码，分别扫描篇幅不同的论文样例进行测试，查看系统运行用时

测试步骤：

1）在程序中添加计时代码

2）把待检查的三个篇幅不同的论文样例和测试程序放在同一目录下，第一篇论文篇幅较少，第二篇论文字数中等符合一般的论文写作字数，第三篇论文篇幅较多

3）运行程序，分别以三个论文作为入口参数，查看输出

4）分析结果

预期结果：

系统运行耗时首先和论文篇幅成正比，但对于使用者来说控制在2秒之内比较合适

运行结果：

如图5.3所示

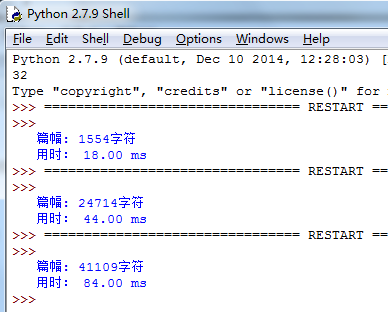


图5.3 测试结果三

结果分析：

尽管python在底层解释执行上速度较低，然而对于使用者来说，对于一篇两万字左右的毕业论文，系统分析的耗时控制在100毫秒之内，远低于体验感良好的控制时长2秒，系统运行的速度完全令人满意。

# 结论

本文首先介绍了毕业论文对于学生完成本科学业的重要性，然后从重要性以及论文本身的科学和严谨性引申到毕业论文格式检查的重要性，从而提出问题，设计一个自动检查格式系统的必要性。接下来本文分析了目前国内在该方面的研究情况，并以这些研究工作为基础，设计出一套可行的方案。

本文侧重从程序运行角度分析系统实现的过程，详细而具体的介绍了如何读取word文档中的xml文件，如何定位一段文本在论文内容中的位置，如何从xml中获取一段文本对应的格式信息，解决了系统在实现过程中的一些关键性技术难点。

然而，本文还存在许多不足之处。由于学生在撰写毕业论文时会出现各种情况的复杂性，系统在实际运行过程中还存在许多不尽人意的地方。

鉴于本文研究工作的不足，在未来的工作中将从如下几方面着手，使研究成果更加完善，促进论文格式检查系统在实际中的应用。

一是进一步分析Microsoft word文档的xml标签定义以及如何组织数据结构，很多有用的信息都隐藏在这些标签里，比如标题级别就有对应的节点标签；进一步分析与字体、段落属性等格式有关的信息如何体现在documen.xml文件和styles.xml文件中，提高系统获取文本格式的准确性。

二是通过大量的具体的测试实例收集问题，将学生撰写毕业论文时有可能出现的各种格式问题都收集起来并编写相应的分析代码，以提高系统定位论文内容的准确性，提高系统的健壮性和实用性。

目前，格式检查系统在实际应用中还未正式上线，未来如果能够更加完善，将会给实际工作带来很大的应用价值。

# 致谢

经过几个月的努力，毕业论文终于撰写完成，毕业设计也以论文作为收尾而结束。在项目的进行和论文的写作过程中，我遇到了很多的困难，但都在老师和同学的帮助下一个个解决了这些问题，最终得以完成毕业设计的程序实现和毕业论文的写作。

首先，我要感谢指导老师李莹，她在系统的设计思路和论文的写作上都给我提供了无私的指导和帮助，指导我顺利完成了毕业论文的撰写。同时在系统设计和实现的过程中，我碰到若干次技术上的难题，每次开组会的时候老师都会针对我项目中出现的问题，为我的项目提出了可行的方案。其次，我要感谢实验室里和我在一个小组的同学们，感谢他们每次讨论时都帮我提出相关的解决方案和建议，由于他们的协助，本系统才能更加顺利的实现。他们也为本系统寻找了数个关键性技术错误，提出了重要的改进意见。我也要感谢与我一起进行毕业设计的同学们。感谢他们为我解决了论文在格式、内容上的一系列琐碎问题。这些问题看似细小，但对论文的整体质量有着重大的影响。

最后，由于我个人学术水平有限，所写论文难免有不足之处，恳请各位老师批评指正！

# 参考文献

[1] 徐东风，彭红星，廖俊杰，基于JAVA的文档格式检查技术的研究极其应用[D]，华南农业大学，2010.

[2] 王杏龙,毛永江.本科毕业论文质量下降的原因及对策[J].高等农业教育,2008(9):69-71.

[3]张印，基于VSTO的高校毕业论文格式化处理软件的设计与实现，[D]北京邮电大学，2012.

[4] 林雪云，基于论文格式智能检查系统的研究，[D].福建师范大学，2009.

[5]王帅群，夏斌，孔薇：基于.NET的论文格式自动检查系统，上海海事大学信息学院[D].，2010.

[6] 阚运奇，毕业论文格式检测系统的设计与研究，东北电力大学，2012.

[7] 陈国胜，何宗明．基于XML技术的Word文档录入及格式检测系统设计[J]．计算机时代，2009(4)：35．37.

[8] 陈呈超，秦勃．基于学术论文质量模型的检索排序算法研究[J]．中国海洋大学学报，2008.

[9] 张卫丰,周国强.毕业设计文档管理技术探索及实践[J].教育与教学研究,2009,11:58-60+67.

[10] 许海洋,李庆,唐平,高明琴.学位论文排版规范性自动检测系统设计[J].广东工业大学学报,2009,04:54-57.

[11] Virantha Ekanayake，Reading and writing Microsoft Word docx files with Python，Posted By Virantha On Fri 16 August 2013.

[12]Magnus Lie Hetland,Beginning Python From Novice to Professional,人民邮电出版社，2014.6.