摘要

1. 引言
   1. 研究背景及意义

随着社会信息化程度的不断提高，办公自动化在这个过程中扮演着重要的角色，而对Office 办公软件的使用又是实现办公自动化的重要组成部分。正因如此，当前大多数高校都会给非计算机专业的大一新生开设一门《计算机应用基础》的课程，在这门课程中就包含了对Office办公软件的操作教学。目前对学生所做Office操作题的批改都是由老师手工进行，这项工作既耗时又容易由于主观因素出现批改标准不一致的情况，为此很有必要开发一款能够自动批改Office操作题的软件，将老师从繁杂的手工批改中解脱出来，实现自动批改。

* 1. 国内外研究现状

目前，自动批改功能大致通过以下四种思路来实现：

构造一个模拟考试环境的仿真系统，并且记录考生的操作步骤，然后通过考生的操作步骤来进行自动阅卷。

采用Office中的VBA技术，通过对Office文档中对象的属性进行分析，并与设置好的的标准答案进行对比，最后得出分数。

通过将文档格式转换为其他格式，如RTF进行处理。

利用一些成熟的操作Office文档的API，采用“一题一编”的形式，对每一套不同的操作题都专门编写一个程序来进行批改。

* 1. 主要研究内容

本论文的研究目标是实现一个基于C#的动态链接库（DLL）以供其它程序使用，该DLL可以实现对Office文档的自动批改。

研究内容包括以以下几个方面：

判断哪些操作可以实现自动批改。

定义考查点和对应的分值。

定位题目的考查点。

获取Office文档中的各个对象的属性并进行分析。

制定合理的阅卷标准。

* 1. 主要难点及解决办法

1. Office 文档结构特点与文档信息获取方法
   1. Office Word文档结构特点
   2. Office Word文档信息获取方法

在Open Xml SDK中，WordprocessingDocument类代表了Word 文档中的一个包。要处理一份Word文档，首先需要通过文档创建一个WordprocessingDocument类的实例，然后操作这个实例从而实现对Word文档的操作。获取该实例之后，我们就能够访问包含了文本的主文档部分。每一个打开的XML包都包含若干个部分，一个WordprocessingDocument至少包含了一个主文档部分，该部分作为一个容器用来容纳Word文档中的主要文本内容，该WordprocessingDocument所代表的包也可以包含其他的一些部分。在一个Word文档中，主文档部分中的文本内容在XML包中使用WordprocessingML来标记。

要从一份Word文档创建WordprocessingDocument类的实例，我们可以调用Open（Stream，Boolean）方法，Open XML SDK中提供了好几个Open方法，我们采用具有2个参数的Open方法来获取实例，第一个参数代表一个Stream，该Stream关联了我们想要打开的Word文档，第二个参数可以取值为true或者false，代表我们打开的Stream是否是可编辑的。在我们的应用当中，由于对Word文档都是只读的，所以该参数均取值为false，表示不能修改打开的Word文档。创建WordprocessingDocument类实例的示例代码如下：WordprocessingDocument document = WordprocessingDocument.Open(stream, false);

要访问主文档部分的body，我们可以把存在的文档体赋值给Body引用，像如下代码所示：

WordprocessingDocument document = WordprocessingDocument.Open(stream, false);

Body body = document.MainDocumentPart.Document.Body;

操作段落

WordprocessingML中的段落

WordprocessingML中最基本的块级内容单元就是段落，段落用<p>元素进行存储。一个段落定义了一个由新行开始的独立内容区域。段落可以包含三部分信息：可选的段落属性、内联内容（比如run）,以及一些可选的校正ID，用来比较2篇文档的内容。

一个段落的属性是通过<pPr>元素指定的，段落属性包含对齐方式、边框、缩进、行距、底纹、文本方向等。下表列出了Open XML SDK中与段落操作相关的常用类与WordprocessingML元素的对应关系：

|  |  |
| --- | --- |
| WordprocessingML元素 | Open XML SDK类 |
| p | Paragraph |
| pPr | ParagraphProperties |
| r | Run |
| t | Text |

Paragraph类

Open XML SDK中的Paragraph类代表为WordprocessingML文档定义的Open XML文件格式schema中的段落（<p>）元素，使用Paragraph对象来操作WordprocessingML文档中的<p>元素。

ParagraphProperties类

在WordprocessingML中，一个段落的属性是通过段落属性（<pPr>）元素指定的，Open XML SDK中的ParagraphProperties类代表<pPr>元素。

Run类

一个Word文档中的段落一般都会包含文本，在WordprocessingML文档的Open XML 文件格式schema中，run（<r>）元素被用来界定一个文本区域。在Open XML SDK中，Run类代表<r>元素。

Text类

在<r>元素内部，是文本(<t>)元素，文本元素是组成文档内容文本的容器。在Open XML SDK中，Text类代表<t>元素。

以下代码实例化了一个Open XML SDK中的Paragraph类对象并使用该对象添加一段文本到WordprocessingML文档中：

public static void WriteToWordDoc(string filepath, string txt)

{

// 使用指定的文件路径来打开一个文档并对其进行编辑

using (WordprocessingDocument wordprocessingDocument =

WordprocessingDocument.Open(filepath, true))

{

// 实例化body对象

Body body = wordprocessingDocument.MainDocumentPart.Document.Body;

// 在一个段落中添加一段文本

Paragraph para = body.AppendChild(new Paragraph());

Run run = para.AppendChild(new Run());

run.AppendChild(new Text(txt));

}

}

上述代码执行完之后，写入WordprocessingML文档中的XML如下所示：

<w:p>

<w:r>

<w:t>测试文本</w:t>

</w:r>

</w:p>

操作表格

在WordprocessingML中另一个类型的块级内容就是表格，一个表格就是一系列排列在行和列中的段落（另一种块级内容）集合。

WordprocessingML中的表格是通过<tbl>元素定义的，该元素与HTML中的<table>标签很相似。表格元素指定了一个表格在文档中呈现的位置。

一个<tbl>元素有两个元素定义了它的属性：<tblPr>和<tblGrid>。其中<tblPr>定义了表格范围（例如样式和宽度）内的属性集合，而<tblGrid>定义了表格的网格布局。一个<tbl>元素可以包含任意非0的行数，每一行通过一个<tr>元素进行指定，每一个<tr>元素可以包含任意非0数目的单元格，每个单元格用<tc>元素进行指定。

下面的表格列出了处理表格时最常用的Open XML SDK中的类

|  |  |
| --- | --- |
| XML元素 | Open XML SDK类 |
| gridCol | GridColumn |
| tblGrid | TableGrid |
| tblPr | TableProperties |
| tc | TableCell |
| tr | TableRow |

Table类

Open XML SDK中的Table类代表为WordprocessingML文档定义的Open XML文件格式schema中的表格（<tbl>）元素，使用Table对象来操作WordprocessingML文档中的< tbl >元素。

TableProperties类

在WordprocessingML中，一个表格的属性是通过表格属性（<tblPr>）元素指定的，Open XML SDK中的TableProperties类代表<tblPr>元素。<tblPr>元素定义了表格级别的属性，使用TableProperties对象可以用来设置WordprocessingML文档中表格的属性。

TableGrid类

Open XML SDK中的TableGrid类代表为WordprocessingML文档定义的Open XML文件格式schema中的表格网格（<tblGrid>）元素，与网格列（<gridCol>）子元素一起，<tblGrid>元素定义了表格的列，并且指定了表格列中单元格的默认宽度，可以使用TableGrid对象来操作WordprocessingML文档中表格中的列。

GridColumn类

Open XML SDK中的GridColumn类代表为WordprocessingML文档定义的Open XML文件格式schema中的网格列（<gridCol>）元素。<gridCol>元素是<tblGrid>元素的子元素，它定义了WordprocessingML文档中一个表格的单个列。可以使用GridColumn类来操作WordprocessingML文档中的单独的列。

TableRow类

Open XML SDK中的TableRow类代表为WordprocessingML文档定义的Open XML文件格式schema中的表格行（<tr>）元素。<tr>元素定义了WordprocessingML文档中表格的一行，与HTML中的<tr>标签相似。一个表格的行也能够使用表格行属性（<trPr>）元素将一些样式应用到它上面。Open XML SDK中的TableRowProperties类代表<trPr>元素。

TableCell类

Open XML SDK中的TableCell类代表为WordprocessingML文档定义的Open XML文件格式schema中的表格单元格（<tcc>）元素。<tc>元素定义了WordprocessingML文档中表格单元格，与HTML中的<td>标签相似。一个表格的单元格也能够使用表格单元格属性（<tcPr>）元素将一些样式应用到它上面。Open XML SDK中的TableCellProperties类代表<trPr>元素。下面的代码展示了如何向一个文档中插入一个1行3列的表格：

public static void InsertTableInDoc(string filepath)

{

// 打开一个Word文档，并使该文档能够编辑

using (WordprocessingDocument wordprocessingDocument =

WordprocessingDocument.Open(filepath, true))

{

// 获取文档体

Body body = wordprocessingDocument.MainDocumentPart.Document.Body;

// 创建一个表格

Table tbl = new Table();

// 设置表格的样式和宽度

TableProperties tableProp = new TableProperties();

TableStyle tableStyle = new TableStyle() { Val = "TableGrid" };

// 使表格宽度占页面宽度的100%

TableWidth tableWidth = new TableWidth() { Width = "5000", Type = TableWidthUnitValues.Pct };

// 将设置的样式应用到表格

tableProp.Append(tableStyle, tableWidth);

tbl.AppendChild(tableProp);

// 添加3列到表格

TableGrid tg = new TableGrid(new GridColumn(), new GridColumn(), new GridColumn());

tbl.AppendChild(tg);

// 创建表格的一行

TableRow tr1 = new TableRow();

// 添加单元格到行中的每一列

TableCell tc1 = new TableCell(new Paragraph(new Run(new Text("1"))));

TableCell tc2 = new TableCell(new Paragraph(new Run(new Text("2"))));

TableCell tc3 = new TableCell(new Paragraph(new Run(new Text("3"))));

tr1.Append(tc1, tc2, tc3);

// 添加行到表格上.

tbl.AppendChild(tr1);

// 添加表格到文档

body.AppendChild(tbl);

}

}

以上代码执行之后会将如下的XMl写入WordprocessingML文档中：

<w:tbl>

<w:tblPr>

<w:tblStyle w:val="TableGrid" />

<w:tblW w:w="5000"

w:type="pct" />

</w:tblPr>

<w:tblGrid>

<w:gridCol />

<w:gridCol />

<w:gridCol />

</w:tblGrid>

<w:tr>

<w:tc>

<w:p>

<w:r>

<w:t>1</w:t>

</w:r>

</w:p>

</w:tc>

<w:tc>

<w:p>

<w:r>

<w:t>2</w:t>

</w:r>

</w:p>

</w:tc>

<w:tc>

<w:p>

<w:r>

<w:t>3</w:t>

</w:r>

</w:p>

</w:tc>

</w:tr>

</w:tbl>

* 1. Office Excel文档结构特点

SpreadsheetML 文档的文档结构包括 <workbook> 元素，后者包含引用工作簿中的工作表的 <sheets> 和 <sheet> 元素。将为每张工作表创建单独的 XML 文件。这些元素是有效电子表格文档所需的最小元素。此外，电子表格文档可能包含 <table>、<chartsheet>、<pivotTableDefinition> 或其他与电子表格相关的元素。

通过使用 Open XML SDK 2.5 for Office，可以使用 SpreadsheetML 元素所对应的强类型类创建文档结构和内容。可以在 DocumentFormat.OpenXML.Spreadsheet 命名空间中找到这些类。下表列出了一些重要电子表格元素所对应类的类名称。

|  |  |  |  |
| --- | --- | --- | --- |
| 包 | SpreadsheetML元素 | Open XML SDK 类 | 描述 |
| 工作簿 | <workbook> | Workbook | 主文档部件的根元素。 |
| 工作表 | <worksheet> | Worksheet | 表示包含文本、数字、日期或公式的单元格网格的工作表类型。 |
| 图表工作表 | <chartsheet> | Chartsheet | 表示存储为自己的工作表的图表的工作表。 |
| 表 | <table> | Table | 指定属于单个数据集的一系列数据的逻辑构造。 |
| 数据透视表 | <pivotTableDefinition> | PivotTableDefinition | 显示可理解布局中数据的聚合视图的逻辑构造。 |
| 透视缓存 | <pivotCacheDefinition> | PivotCacheDefinition | 定义数据透视表中的数据源的构造。 |
| 透视缓存记录 | <pivotCacheRecords> | PivotCacheRecords | 数据透视表的源数据的缓存。 |
| 计算链 | <calcChain> | CalculationChain | 指定上次计算工作簿中单元格的顺序的构造。 |
| 共享字符串表 | <sst> | SharedStringTable | 包含每个唯一字符串在工作簿中的所有工作表上的出现次数的构造。 |
| 条件格式 | <conditionalFormatting> | ConditionalFormatting | 定义应用于一个单元格或一系列单元格的格式的构造。 |
| 公式 | <f> | CellFormula | 定义包含公式的单元格的公式文本的构造。 |

* 1. Office Excel文档信息获取方法
  2. Office PowerPoint文档结构特点
  3. Office PowerPoint文档信息获取方法

1. Word 文档结构分析
2. Excel 文档结构分析
   1. 演示文稿部件

[**最小演示文稿文件的结构**](javascript:void(0))

我们可以使用 Open XML API 逐个部件构建最小演示文稿文件。

最小演示文稿文件包含一个由文件presentation.xml表示的演示文稿部件以及一个演示文稿属性部件(presProps.xml)、一个幻灯片母版部件 (slideMaster.xml)、一个幻灯片版式部件(slideLayout.xml)和一个主题部件 (theme.xml)。有一个或多个幻灯片部件(slide.xml)是可选的。

演示文稿文档的打包结构包含各部件之间的多个引用，其中包括一些循环引用。例如，幻灯片版式引用幻灯片母版，幻灯片母版引用幻灯片版式。

[生成的PresentationML XML代码](javascript:void(0))

在运行 Open XML SDK 2.5 代码生成演示文稿之后，可以了解 .zip 包的内容，以查看 PresentationML XML 代码。若要查看 .zip 包，可将最小演示文稿的扩展名由 .pptx 重命名为 .zip。在 .zip 包中，有多个构成最小演示文稿的部件。

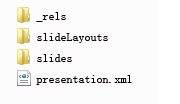
图 1 显示 .zip 包的 ppt 文件夹下用于包含单个幻灯片的最小演示文稿的结构。

图 1. 最小演示文稿文件夹结构

presentation.xml 文件包含引用演示文稿中的幻灯片的 <sld> （幻灯片）元素。每个幻灯片通过幻灯片 ID 和关系 ID 与演示文稿关联。slideID 是在包中用于标识幻灯片的标识符 (ID)，在演示文稿中必须唯一。id 属性是标识与幻灯片关联的幻灯片部件定义的关系 ID。

下面的 XML 代码为 PresentationML，它表示包含单个幻灯片的演示文稿文档的演示文稿部件。此代码在您运行 Open XML SDK 2.5 代码以创建最小演示文稿时生成。

XML

<?xml version="1.0" encoding="utf-8"?>

<p:presentation xmlns:p="http://schemas.openxmlformats.org/presentationml/2006/main">

<p:sldMasterIdLst>

<p:sldMasterId id="2147483648"

r:id="rId1"

xmlns:r="http://schemas.openxmlformats.org/officeDocument/2006/relationships" />

</p:sldMasterIdLst>

<p:sldIdLst>

<p:sldId id="256"

r:id="rId2"

xmlns:r="http://schemas.openxmlformats.org/officeDocument/2006/relationships" />

</p:sldIdLst>

<p:sldSz cx="9144000"

cy="6858000"

type="screen4x3" />

<p:notesSz cx="6858000"

cy="9144000" />

<p:defaultTextStyle />

</p:presentation>

下面的 XML 代码是表示演示文稿文档的关系部件的 PresentationML。此代码在您运行 Open XML SDK 2.5 以创建最小演示文稿时生成。

XML

<?xml version="1.0" encoding="utf-8"?>

<Relationships xmlns="http://schemas.openxmlformats.org/package/2006/relationships">

<Relationship Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/slide"

Target="/ppt/slides/slide.xml"

Id="rId2" />

<Relationship Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/slideMaster"

Target="/ppt/slideLayouts/slideMasters/slideMaster.xml"

Id="rId1" />

<Relationship Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/theme"

Target="/ppt/slideLayouts/slideMasters/theme/theme.xml"

Id="rId5" />

</Relationships>

下面的 XML 代码是表示演示文稿文档的幻灯片部件的 PresentationML。演示文稿中的每个幻灯片都有与之关联的幻灯片部件。此代码在您运行 Open XML SDK 2.5 以创建最小演示文稿时生成。

XML

<?xml version="1.0" encoding="utf-8"?>

<p:sld xmlns:p="http://schemas.openxmlformats.org/presentationml/2006/main">

<p:cSld>

<p:spTree>

<p:nvGrpSpPr>

<p:cNvPr id="1"

name="" />

<p:cNvGrpSpPr />

<p:nvPr />

</p:nvGrpSpPr>

<p:grpSpPr>

<a:xfrm xmlns:a="http://schemas.openxmlformats.org/drawingml/2006/main" />

</p:grpSpPr>

<p:sp>

<p:nvSpPr>

<p:cNvPr id="2"

name="Title 1" />

<p:cNvSpPr>

<a:spLocks noGrp="1"

xmlns:a="http://schemas.openxmlformats.org/drawingml/2006/main" />

</p:cNvSpPr>

<p:nvPr>

<p:ph />

</p:nvPr>

</p:nvSpPr>

<p:spPr />

<p:txBody>

<a:bodyPr xmlns:a="http://schemas.openxmlformats.org/drawingml/2006/main" />

<a:lstStyle xmlns:a="http://schemas.openxmlformats.org/drawingml/2006/main" />

<a:p xmlns:a="http://schemas.openxmlformats.org/drawingml/2006/main">

<a:endParaRPr lang="en-US" />

</a:p>

</p:txBody>

</p:sp>

</p:spTree>

</p:cSld>

<p:clrMapOvr>

<a:masterClrMapping xmlns:a="http://schemas.openxmlformats.org/drawingml/2006/main" />

</p:clrMapOvr>

</p:sld>

第6章 自动评阅程序中的一些策略研究

* 1. 考点类型划分策略

word操作题考点类型划分：

根据对大量Word操作题的题干进行分析的结果，将Word操作题考点类型分为以下十一个大类，

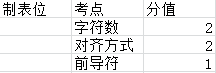
段落、艺术字、表格、文本、图片、分栏、首字下沉、SmartArt、文本位置、文档排版、制表位。其中，每个大类所覆盖的小类如下表格所示：

|  |  |
| --- | --- |
| 大类 | 所含小类 |
| 段落 | 边框类型、边框颜色、边框粗细、边框阴影、底纹 |
| 艺术字 | 形状样式、文字样式、位置 |
| 表格 | 表格样式、表格边框、表格底纹、表头对齐、表头字体、其他对齐、其他字体 |
| 文本 | 大小、颜色、加粗、删除线、下划线、斜体、字体、对齐方式 |
| 图片 | 图片大小、图片样式 |
| 分栏 | 分栏数、分隔线、间隔宽度 |
| 首字下沉 | 下沉/悬挂、下沉行数、距正文 |
| SmartArt | 布局、颜色、样式、文字 |
| 文本位置 | 提升/降低 |
| 文档排版 | 段落首行缩进、段落间距 |
| 制表位 | 字符数、对齐方式、前导符 |

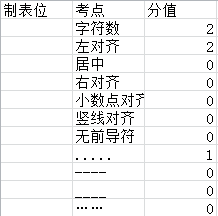
在考点的类型划分上面，若划分粒度过大，则会使批改操作产生较大误差，若划分粒度过小，则会给系统的开发以及使用带来较大的复杂性，因此在该系统中采取了一种适中的粒度，既能防止粒度过大导致的较大误差，又能减轻粒度过小所带来的复杂性。以“制表位“这个大类来进行说明：

若考点划分成“制表位”这个一个大类，则对于该考点，要么得满分，要么得零分，在这个考点上无法进行更细致的划分，因此会产生较大误差。若将制表位的对齐方式考点再细化成“左对齐”、“居中”、“右对齐”、“小数点对齐”、“竖线对齐”，将前导符划分成“无前导符”、“…..”、“——”、“\_\_\_\_”、“……”这五种前导符类型，虽然在批改上面可以达到非常精确的匹配，但是在系统的实现上，以及配置文件的设置上面会大大增加复杂度。

下图显示了制表位粒度适中、粒度过细在配置文件中的体现，我们可以直观的看出这2种粒度的复杂度：粒度过细的复杂度是适中粒度的将近4倍。要准确无误地配置出来这样一个细粒度配置文件，对系统的使用者来说是一个极大的挑战，与要减轻老师批改负担这一目标不符，鉴于此，在该系统的实现上，采用了适中粒度的考点类型划分策略。



适中粒度



粒度过细

* 1. 考点位置定位策略

策略一：根据字符串进行定位

对于考点涉及到具体文本的

* 1. 考点信息匹配策略
  2. 评阅结果生成策略

第7章 自动评阅程序的实现

* 1. 整体设计思路

第一步：定义考点类型以及对应考点所占分值的文件格式。

第二步：读取第一步中的文件，建立“考点-分值”模型。

第三步：读取标准答案文件，根据“考点-分值”模型中的考点找出标准答案文件中相应属性的值建立“考点-属性值”模型。

第四步：读取要批改的文件，根据“考点-分值”模型中的考点找出要批改文件中相应的属性值，将该属性值与“考点-属性值”模型中的属性值进行对比，若一致，则获取该考点对应的分值。

* 1. Word评阅程序设计思路及主要实现代码
  2. Excel评阅程序设计思路及主要实现代码
  3. PowerPoint评阅程序设计思路及主要实现代码

第8章 总结与展望

* 1. 全文小结
  2. 对未来的展望

# 总结与展望

**6.1 工作总结**

随。着互。联网的发展，“互联网+”已经成为这个时代的主旋律，

**6.2 研究展望**

Office主观题自动评阅系统虽然已经得到了广泛的关注与研究，但在技术上仍然需要不断地完善。无论是考点类型划分还是考点定位策略等都值得深入研究。本文采用Open XMLSDK对Office文档信息进行提取，并通过选取适中粒度的考点类型配置方式，在一定程。度上改。善了Office主观题自动评阅系统的适用范围和系统复杂度，同时也取得了良好的性能和效果，但该系统也存在一些不足之处，未来可以考虑从如下几个方面进行进一步的研究：

(1) 在考点类型划分上面需要进一步调研，既要控制好考点类型划分的粒度，同时也要尽量避免各个考点之间产生重叠，使得考点之间会相互影响，从而使最终的评阅结果产生较大的误差。

(2) 考点定位策略的选取是该系统实现过程当中非常关键的一步，如何保证该定位策略能够定位到要批改的考点，以及对于同一个考点考查多处的情况，我们该如何进行区分还有很多值得我们探索的地方。

(3) 人工批改与程序自动批改的一个区别在于：若整个文档产生了较小的变动，在程序的视角来看可能变化是巨大的，而人工看起来则可能意识不到这种变化，从而使得两者最终批改结果有较大误差。这就需要进一步完善系统，在一些容易由于学生误操作的地方采取一些措施，将这种误差降到最低。