文章编号：2096-1472(2021)-05-30-03
DOI:10.19644/j.cnki.issn2096-1472.2021.05.008

基于报表和模板的报告生成方法研究

毛燠锋，潘玉春，朱玉付

(南京南瑞继保电气有限公司，江苏 南京 211102)
✉maoyf@nrec.com; panyc@nrec.com; zhuyufu@nrec.com

摘　要：随着电力报表在输变电运维中的普及以及数据分析报告生成的迫切需求，急需一种将报表数据转换成可用的分析报告的方法。本文针对上述需求，提出了一套完整的基于电力报表和Apache POI技术的分析报告生成流程，实现了将历史数据以图表的形式嵌入预设模板中，并最终形成格式统一的分析报告，有效地提高了数据分析人员的效率。

关键词：数据挖掘；电力报表；分析报告；Apache POI；模板
中图分类号：TP391　**文献标识码**：A

A Report Generation Method based on Report Form and Template

MAO Yufeng, PAN Yuchun, ZHU Yufu

(*NR Electric CO., LTD., Nanjing* 211102, *China*)
✉maoyf@nrec.com; panyc@nrec.com; zhuyufu@nrec.com

Abstract: With the popularity of power reports in operation and maintenance of power transmission and transformation, and the urgent need for data analysis reports, a method of converting report data into usable analysis reports is urgently needed. To meet the above needs, this paper proposes a complete analysis report generation process based on power reports and Apache POI technology. The proposed method makes historical data embedded into the preset template in the form of charts, and finally forms a unified analysis report. It effectively improves the efficiency of data analysts.

Keywords: data mining; power report; analysis report; Apache POI; template

1 引言(Introduction)

虽然电力行业目前有很多成熟的B/S或者C/S数据展示系统，但通常是作为展示工具的，即使有下载功能也仅仅是将查询到的数据以各种图表的形式展示出来，无法直接生成分析报告的形式。对于需要分析报告用于汇报或者归档的用户来说，实现分析文档的自动生成工作必要而且急切。本文介绍了一种方便快捷的分析文档生成方式，即先按照用户的需求使用报表工具将电力实时库中的数据以图表的形式导出到Excel中，然后使用POI类库取出相应的数据并插入预设模板中生成符合用户需求的分析报告。针对生成的报告在不同的Office工具中格式不一致问题，使用OpenOffice[1]软件实现版本转换，并最终生成样式固定的可直接使用的数据分析报告。

2 相关技术(Related technologies)

电力数据通常保存在实时数据库中，要利用相关数据生成文档需要将数据库中的数据提取出来并以图表的形式进行表示，鉴于电力报表的高实用性与可扩展性，可用于实现该功能。在将需要的数据导出成图表等格式后，需要将其插入预设文档中。POI类库是Java中操作文档的通用性工具类，在其基础上扩展可形成用于文档内容插入的基础类，使用该类可实现图表数据到模板数据的转化。在得到分析文档后由于使用了合并表格单元格操作，导致在不同的文档工具中表格的预览效果出现差异，可以使用开源的OpenOffice进行转化从而实现格式一致，在用户检查无误后该分析文档可作为最终的分析报告进行发布。

2.1 电力报表

报表是以表格、图表的形式来动态展示数据的可视化工具[2]，使用报表工具可以对数据库中的各类数据进行实时分析，进而用于辅助管理决策。报表工具具有专业、简捷、灵

活的特点，在部署设置完成后可以直接使用，形成基于特定领域的数据分析展示系统。使用纯Java开发的报表工具结合具体的电力行业业务需求，便形成了电力报表工具，在电力报表的基础上结合具体的数据和流程便形成了电力数据分析子系统。由于使用的报表工具是纯Java开发，对于不满足需求的功能或者新增功能可以进行二次开发，在此基础上形成满足实际需求的电力报表工具。

本文在电力报表的基础上将需求的数据导出到分类Excel中，包括告警、支流、环流、运行单位等统计信息。导出格式按统计信息的类型分布在不同的Excel中，对于相同类型但层次不同的数据可以分布在不同的sheet中，Excel中格式需要保持稳定，方便后续程序进行遍历读取处理。

2.2 POI类库

Apache POI是Apache软件基金会开发的标准类库[3]，具有使用方便、集成度高、可扩展性强等特点。在Java程序中使用POI类库可实现对Microsoft Office进行各种基本操作，包括生成文档文件、提取文档信息和修改样式等。本文中主要使用POI技术实现从Excel中读取图表数据并按照规范名称存放，后续再将其插入分析模板中并最终形成分析文档。具体操作流程图如图1所示。

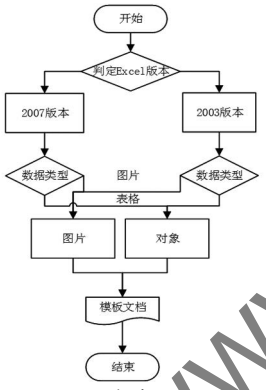


图1 POI解析流程图

Fig.1 POI analysis flow chart

2.2.1 图表提取

Apache POI类库是将Excel中各个元素(包括Workbook、sheet、row、cell等)的操作方法封装成函数的形式[4]，开发者在POI类库的基础上使用自定义函数将Excel各个元素中包含的信息提取出来。

在使用报表工具将需要的图表导出到Excel后，可以使用POI类库将其提取出来并分类保存，对于图片格式数据按规范名称保存在文件夹下，对于表格以对象的形式保存在内存中，方便后续使用。

算法1 使用POI提取图表信息

输入：表格的工作簿(HSSFWorkbook)workbook，outImgPath图片保存的位置

```
1：function getSheetPictrues(workbook)
2：SSFPictureData > pictures = workbook.getAllPictures();//获取所有图片信息
3：for HSSFShape shape : sheet.getDrawingPatriarch().getChildren() do
4：    HSSFPicture pic = (HSSFPicture) shape;
5：    int pictureIndex = pic.getPictureIndex() - 1;
6：    HSSFPictureData picData = pictures.get(pictureIndex);
7：    String picIndex = fileNameNoExt+(sheetNum+1)+(pictureIndex+1);//设置图片名称
8：    sheetIndexPicMap.put(picIndex, picData); //将名称和信息以键子对的形式存储在map中
9：  end for
10：end function
11：
12：function getSheetTable(sheet)  //获取表格信息
13：for j = sheet.getFirstRowNum()—>sheet.getPhysicalNumberOfRows()+ sheet.getFirstRowNum() do
14：  Row row = sheet.getRow(j);//定位到sheet中的一行
15：    Cell cell = row.getCell(k);//定位于单元格
16：    obj = cell.getNumericCellValue();//获取单元格值
17：  rowValue.put(k, obj);//将值保存到内存中
18：  end for
19：end function
```

上述操作可以将报表保存到Excel中的数据提取出来，对于图片信息按照自定义的名称保存在本地，对于表格信息以map键子对的形式提取到内存中。在对图片和表格提取完成后，准备工作已经就绪。

2.2.2 图表渲染

poi-tl是在Apache POI的基础上纯Java开发的自定义Word模板引擎，具有跨平台、易移植、集成度高等特点。使用poi-tl技术可以实现在预设模板的基础上插入信息，并保持格式一致性，即将模板中的{{info}}等信息替换成需要的信息，其中{{}}为通配符，info可为具体信息，也可为模块信息(表格、图片等)，根据用户的实际需求设置即可。

插入表格的代码流程如下：

算法2 使用poi-tl实现图表渲染

输入：tableList表格数据对象

```
1：function renderTable(tableList)
2：  for int i:tabIndex.size() do  // tabIndex.size()为表格宽度
3：    textRender.add(new TextRenderData(color, String.valueOf(tableList1.get(1).get(i))));//设置表头
4：  end for
5：  header = new RowRenderData(textRender, BackgroundColor);//渲染表头
6：  rows = RowRenderData.build(row);//row为行数据
7：  tableDatas.add(rows);//加入数组中
```

```
8： test=new MiniTableRenderData(header, tableDatas, MiniTableRenderData.WIDTH_A4_FULL);//表格数据渲染
9： data1.put(tab, test);//生成表格
10： data1.put(picName,new PictureRenderData(550, 350,pictures[i].getName()));//生成图片
11： Configure config=Configure.newBuilder().customPolicy("hl0",new DetailTablePolicy()).build();
12： XWPFTemplate template = XWPFTemplate.compile(moban, config);//将规则应用到模板中
13： template.render(data1);//将data1数据整个插入文档中
14： FileOutputStream out = new FileOutputStream(outDoc);//输出生成的报告
15：end function
```

预设模板是将用户需要的文档中的具体内容替换成通配符，完成预设模板的设置后可直接运行程序提取信息并插入模板。针对生成文档中需要调整的地方，可以将需求提供给开发人员进行相应的程序调整，对于无法调整的则需要用户进行手动调整并最终形成用户需要的报告形式。对于需要合并单元格的不规则表格，可以使用key值匹配自定义规则DetailTablePolicy进行操作，最终得到样式丰富的表格样式。针对表格数据的陈述性描述，则可以在程序中进行组织并插入通配符中，图片数据在设置大小后也会匹配到相应的位置。经过上述步骤可以实现将报表中导出的数据完美适配到模板文档中，并经过调整获取最终用户需要的报告的形式。

2.3 OpenOffice转换

实验的过程中会遇到生成的Word分析文档中的表格在不同的工具(Microsoft Office和WPS等)下展示的格式会发生偏差，这是由于不同的Office中定义的样式不一致导致的。为了保持格式一致，必须将样式统一，在实验的过程中发现使用Microsoft Office的版本转换工具可以使样式保持统一，由于版权问题最终选择使用OpenOffice来实现相应的功能。OpenOffice是一款优秀而且免费的跨平台办公软件，具有安装方便、功能强大等特点[5]，与目前主流的办公软件也兼容。使用OpenOffice可以实现Office的在线编辑和格式转换等工作，该软件提供统一的对外接口，开发者可以根据实际需求使用命令调用相应的功能，但相对的异常处理过程比较烦琐。为了简化操作，在OpenOffice上引入jodconverter架包用于管理OpenOffice。jodconverter是一个文档转换器工具包，在OpenOffice、PageOffice等开源Office工具的基础上实现文档的转换工作，具有集成度高、可操作性强的特点。

Jodconverter是一款纯Java编写的Office文件转换器，实现了对Office工具相关功能的使用和控制[6]，其使用相关的方法实现相应的程序接口调用，用户只需要关注需要实现的功能而不用关注具体的实现和处理过程，具体实现如图2所示。

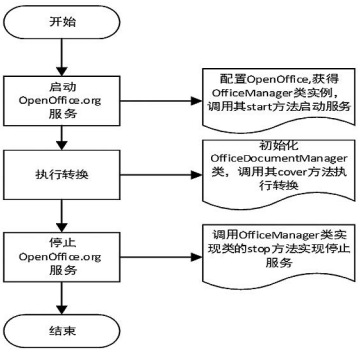


图2 Jodconverter算法流程图

Fig.2 Jodconverter algorithm flow chart

3 实验效果(Experimental effect)

本章在上述技术的基础上使用模板文档生成目标需要的分析文档，在用户检查无误的情况下可直接作为最终可发布的报告。其中模板文档将动态变化的部分用通配符代替，存放在程序部署的服务器中，由于报表导出数据耗时较多，报表的生成使用crontab[7]执行生成脚本作为定时任务，而图表的提取以及模板的插入耗时较少，在用户点击下载文档按钮时执行。在做好上述准备后，即可运行程序生成最终的文档，实验效果如图3和图4所示。


（二）直流停运次数统计

{{yearanseason}}各直流停运总次数统计如表 1.3 所示。

{{hejicishu}}

表 1.3 直流停运总次数统计

{{zhiliutingyuncishu0}}

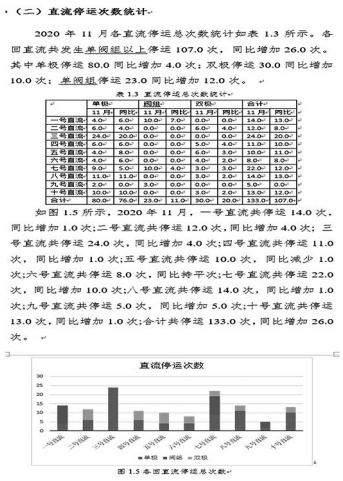
如图 1.5 所示，{{yearanseason}}，{{zongcishu}}

{{@zhiliutingyuncishu11}}

图 1.5 各回直流停运总次数

图3 预设模板

Fig.3 Preset template


（二）直流停运次数统计

2020 年 11 月各直流停运总次数统计如表 1.3 所示。各回直流共发生单阀组以上停运 107.0 次，同比增加 26.0 次。其中单极停运 80.0 同比增加 4.0 次；双极停运 30.0 同比增加 10.0 次；单阀组停运 23.0 同比增加 12.0 次。

表 1.3 直流停运总次数统计

	单极		阀组		双极		合计	
	11 月	同比	11 月	同比	11 月	同比	11 月	同比
一号直流	4.0	6.0	10.0	7.0	0.0	0.0	14.0	13.0
二号直流	6.0	4.0	0.0	0.0	6.0	4.0	12.0	8.0
三号直流	24.0	20.0	0.0	0.0	0.0	0.0	24.0	20.0
四号直流	6.0	6.0	0.0	0.0	5.0	4.0	11.0	10.0
五号直流	4.0	8.0	0.0	0.0	6.0	3.0	10.0	11.0
六号直流	4.0	6.0	0.0	0.0	4.0	2.0	8.0	8.0
七号直流	9.0	5.0	10.0	4.0	3.0	3.0	22.0	12.0
八号直流	11.0	11.0	0.0	0.0	3.0	2.0	14.0	13.0
九号直流	2.0	0.0	3.0	0.0	0.0	0.0	5.0	0.0
十号直流	10.0	10.0	0.0	0.0	3.0	2.0	13.0	12.0
合计	80.0	76.0	23.0	11.0	30.0	20.0	133.0	107.0

如图 1.5 所示，2020 年 11 月，一号直流共停运 14.0 次，同比增加 1.0 次；二号直流共停运 12.0 次，同比增加 4.0 次；三号直流共停运 24.0 次，同比增加 4.0 次；四号直流共停运 11.0 次，同比增加 1.0 次；五号直流共停运 10.0 次，同比减少 1.0 次；六号直流共停运 8.0 次，同比持平次；七号直流共停运 22.0 次，同比增加 10.0 次；八号直流共停运 14.0 次，同比增加 1.0 次；九号直流共停运 5.0 次，同比增加 5.0 次；十号直流共停运 13.0 次，同比增加 1.0 次；合计共停运 133.0 次，同比增加 26.0 次。

图 1.5 各回直流停运总次数

图4 生成报告

Fig.4 Generating reports

从上图可以发现，原始模板中可设置的内容使用通配符进行替换[8]，对于表格信息需要先插入单个单元格并定义好格式，不规则表格的渲染在程序后端定义规则后根据info值进行引入。由于表格信息存放在内存中，对于需要文字渲染的

(下转第21页)