# 项目介绍

目录

[项目介绍 1](#_Toc50340290)

[项目背景——毕业论文排版难 1](#_Toc50340291)

[LaTeX阻力大 1](#_Toc50340292)

[Word不好用 2](#_Toc50340293)

[解决方案 3](#_Toc50340294)

[已有自动化排版方案不能满足需求 3](#_Toc50340295)

[基于pandoc的markdown转换方案 3](#_Toc50340296)

[markdown 3](#_Toc50340297)

[markdown 与 LaTeX和Word 的核心区别 3](#_Toc50340298)

[markdown的优势 4](#_Toc50340299)

[pandoc扩展的markdown 4](#_Toc50340300)

[目前pandoc结合markdown转Word方案的不足 4](#_Toc50340301)

[实现思路 5](#_Toc50340302)

[pandoc插件工作原理 5](#_Toc50340303)

[可行性证明 5](#_Toc50340304)

[具体操作 5](#_Toc50340305)

[技术路线 5](#_Toc50340306)

[编写pandoc插件 5](#_Toc50340307)

[直接编辑修改之后的docx文件 6](#_Toc50340308)

[预期规划 6](#_Toc50340309)

## 项目背景——毕业论文排版难

### LaTeX阻力大

* 众所周知，论文排版通常使用Word和LaTeX，众多知名高校同时提供了这两种方式的支持，且提供了官方的LaTeX模版。
* 而我校虽然身为985高校，却迟迟没有提供对LaTeX的支持，更遑论官方的LaTeX模版。虽然学校要求使用Word输出的PDF提交，但自行使用LaTeX排版的论文可能会因为与Word的实现效果不同而受到“责难”，同时如果被抽到送审查重，是需要提供Word版本的论文的。

因此，虽然LaTeX输出的PDF在文件格式上是“兼容”的，也有前辈做出了LaTeX的模版，但还是需要自行承担排版格式不被认可的风险，哪怕这样，LaTeX也存在

* 学习成本极高，需要较强的动手能力，编程素养和学习能力
* 编译缓慢，每次编译需要花费十秒左右的编译时间
* 缺少相关支持和学习环境，遇到问题难以求助

等问题。

而如果格式不通过，需要学习大量LaTeX格式知识手动调整，也就失去LaTeX的意义了

因此，对于学生而言，使用Word仍然是成本最低最保险的选择。

### Word不好用

但是Word却又面临

* 学校并未提供模版，需要手动反复调整格式
  + 而大部分学生并不具备该能力，排出的论文在格式上杂乱无章
  + 甚至学校给出的排版范例本身的格式也并不符合排版规范
* 大部分学生并不会结合排版规范使用Word的样式等自动化排版功能

即使解决模版问题，仍有

* 公式输入极其繁琐
* 公式编号困难
* Word程序本身容易卡顿不稳定

等问题

因此，如果我们能在Word的基础上，给出一种足够自动化，学习成本也不是很高，并且简单容易的可选替代方案，那么将极大的改进Word论文的写作体验。

## 解决方案

### 已有自动化排版方案不能满足需求

对于照顾低水平用户的Word排版自动化这一问题，已有论述相关排版方案的论文（主要是一些学校计算机学院的本科毕业论文）的方案是，使用各种语言构建图形化操作界面，调用win32接口利用VBA操纵Word，通过简单的查找替换和正则表达式匹配，实现将纯文本编号，替换为Word原生编号，以及匹配标题样式等功能。这样的做法通常需要用户先用“纯文本”的“土办法”把各个编号写好，且对于Word小白排出的各种杂乱格式不能做到完全匹配和疏通，还容易改出新的排版问题。

因此已有的论文方案并不能很好的改善Word排版自动化难的问题。

### 基于pandoc的markdown转换方案

pandoc是一个开源的“通用的文档转换器”，可以实现对.docx，.html，.tex，.epub，甚至是.pptx等格式的相互转换。（当然是会有各种各样的格式损失的）

而pandoc的转换核心便是基于该项目采用的markdown格式，关于markdown的篇幅占据了pandoc手册三分之一左右[[1]](#footnote-1)

同时pandoc本身支持插件，可以用插件增强实现在转换中的数据和格式处理。

### markdown

markdown是一门简单的标记语言，通过简单的格式符号即可实现常用的“标题”，“正文”，“列表”，“引用”，“代码块”，“LaTeX数学公式”，“图片”，“表格”等内容的表示。

TODO 图片展示

### markdown 与 LaTeX和Word 的核心区别

不管是LaTeX还是Word，都是基于“纸张”的排版模式，更注重的是内容打印在纸张上的“格式”。

而markdown没有任何纸张的概念，markdown也不管任何格式如何“被显示”，只专注于核心的写作内容，即想表达的思想，而不用花时间在与内容无关的格式上，哪怕是一点点！

LaTeX更加擅长高级的如书籍之类的“排版”，对于论文的格式远比各种复杂的书籍简单，用LaTeX排版显得过于臃肿，且TeX发明到现在已经近40年，那个时候连PDF格式都没有，许多由于当时计算机性能不足而产生的设计在如今看来是非常可笑的

Word则可以看作是90年代计算机远未普及，日常生产生活中需要大量依赖纸质文件传达信息，这个时候Word以简单的图形化设计让人人都能轻松制作出较为整齐的轻量化纸质文件，专业性方面又足够可以胜任相对较长的论文。但是如今电子设备大量普及，无纸化办公成为趋势，很多场景已经不需要Word，而Word作为“排版”软件，只能是死板的“白纸黑字”，不支持“夜间模式”，也不能支持灵活的页面宽度调整，已经有“屠龙者变恶龙”的趋势了

### markdown的优势

* markdown是简单的——10分钟的视频或是文字教程即可学会核心语法
* markdown是面向未来的——计算机领域用脚投票出来文档写作语言，越来越多的软件支持markdown语法，未来不支持markdown语法的文字录入工具极有可能被淘汰
* markdown是可扩展的——不同的地方可以基于不同的需求自行扩展相应语法实现不同功能

### pandoc扩展的markdown

pandoc工具本身对markdown进行了语法增强，实现了 **“交叉引用”，“脚注”，“增强的多级列表”** 等功能，进一步扫清了markdown在论文写作中的不足部分。

### 目前pandoc结合markdown转Word方案的不足

网络上已经有一些用markdown来写论文再经由pandoc转换的方案，且用上了相应插件来实现交叉引用等功能，但由于docx格式的特殊性以及pandoc的一些局限，目前没有一整套的方案可以同时实现Word中原生的

* 题注（图片表格的编号）及其交叉引用
* 公式右侧编号
* 分节符，分页符，制表符的插入
* 复杂页眉页脚设置

等功能

## 实现思路

### pandoc插件工作原理

输入格式 –> 抽象得到json数据格式（仅包含文字和对应的样式） –> **将json传入各个插件依次处理** –> 插件处理完之后的新json -> 根据“模版文件”（预设了样式的Word）样式得到输出格式

### 可行性证明

* 不管pandoc能不能直接把markdown转换成符合要求的Word文件（docx格式），最终一定会得到格式标准的docx输出文件
* docx文件采用zip格式压缩封装，可以解压处理后再封装
* docx所基于的ooxml格式在2006年成为国际标准

TODO xml图片

结论：不管pandoc能不能最终完成输出符合论文排版要求的Word，都一定可以通过编程进行后续处理使其符合论文排版要求

### 具体操作

* 自行规定扩充一定的markdown语法来表示markdown原来无法表示的语法，如“制表符”，“分页符”，“分节符”等
* 通过pandoc或其他操作，把增强的语法变成Word中原生的功能

具体详细的需要处理的内容此处不表

## 技术路线

根据两种思路，存在两条实现方案，且可以互为补充共同生效

### 编写pandoc插件

可行性：根据pandoc文档，可以根据特定的语法直接插入Word的ooxml代码，该代码将在转换为Word后直接变成docx文件的一部分。

只需要预设好特定的简化标识，即可利用插件在转换的过程中将简化的标识替换成对应功能的ooxml代码，实现相应功能。

TODO 插入ooxml代码实现分节符

难点：

* 部分功能理论上无法用该方式实现，尝试寻找替代实现方法或使用另一技术路线补充
* 数据结构较为复杂

### 直接编辑修改之后的docx文件

可行性无需怀疑，较为直观，只需要处理xml文件即可实现所有功能

难点：

* 缺少处理xml的工具（python中的xml工具大多用以提取数据或简单新增，实现树状的重组替换较为复杂）
* docx 文档结构较为复杂

## 预期规划

* 技术预研——接近完成
* 程序编写——2021年2月前完成
* 中文文档编写——2021年4月
* 校验测试排错——2021年5月
* 英文文档编写

1. 根据pandoc手册PDF版本的页数 [↑](#footnote-ref-1)