苏州大学实验报告

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 院系 | 计算机学院 | | 年级专业 | | 21计科 | | 姓名 | 方浩楠 | 学号 | 2127405048 |
| 课程名称 | | 编译原理课程实践 | | | | | | | 成绩 |  |
| 指导教师 | | 王中卿 | | 同组实验者 | | 无 | | 实验日期 | 2023.11.7 | |

|  |  |
| --- | --- |
| 实 验 名 称 | 基于PLY的LaTex解析 |

1. 实验目的

理解并实践将LaTeX文档转换为PDF的过程。

掌握解析LaTeX文件并生成抽象语法树（AST）的技术。

学习如何将AST进一步转换为HTML和PDF格式。

提升Python编程和处理文本文件的能力。

1. 实验内容

使用ply库来解析LaTeX文件并创建AST。

通过自定义的Node.py模块建立和操作AST。

使用AST2HTML.py将AST转换为HTML格式。

使用HTML2PDF.py将HTML格式转换为PDF文件。

分析并调试转换过程中的潜在问题。

1. 实验步骤和结果

项目结构如下:

表格

描述已自动生成

项目运行方式:

要运行此项目，需要安装以下依赖项：

**ply~=3.11**

**pytest~=7.4.3**

**fpdf2~=2.7.6**

您可以通过运行以下命令来安装这些依赖项：

**pip install -r requirements.txt**

使用方法

方式1

使用命令行参数指定要转换的文件路径，例如：

**python run.py data/example1.tex output/example1.pdf output/example1.html**

这条指令会读取`data/example1.tex`文件，并将其转换为`output/example1.pdf`和`output/example1.html`文件。

只要是形如

**python run.py {读取的latex文件位置} {输出的pdf文件位置} {输出的html文件位置}**

的指令均可以运行

其中html文件作为中间文件，可以不指定路径.这样就不会生成html文件,而是直接将latex文件转换为pdf文件

方式2

也可以直接打开run.py,然后修改run.py的这一部分:

**if \_\_name\_\_ == "\_\_main\_\_":**

**tex\_filename = ""**

**pdf\_output\_filename = ""**

**html\_output\_filename = ""**

修改这一部分也可以实现转换功能

标签实现情况如下:

|  |  |
| --- | --- |
| \begin{document}...\end{document} | 实现 |
| \title | 实现 |
| \author | 实现 |
| abstract | 实现 |
| \section | 实现 |
| \subsection | 实现 |
| itemize | 实现 |
| item | 实现 |

AST的生成情况:

对于example2.tex:

**+ [DOC]**

**+ [CONTENT]**

**+ [TITLE]**

**+ How to Structure a Latex Document**

**+ [AUTHOR]**

**+ Andrew Roberts**

**+ [ABSTRACT]**

**+ In this article, I shall discuss some of the fundamental topics in producing a structured document. This document itself does not go into much depth, but is instead the output of an example of how to implement structure. Its Latex source, when in used with my tutorial provides all the relevant information.**

**+ [SECTIONS]**

**+ [SECTION](Introduction)**

**+ This small document is designed to illustrate how easy it is to create a well structured document within Latex. You should quickly be able to see how the article looks very professional, despite the content being far from academic. Titles, section headings, justified text, text formatting etc., is all there, and you would be surprised when you see just how little markup was required to get this output.**

**+ [SECTION](Structure)**

**+ One of the great advantages of latex is that all it needs to know is the structure of a document, and then it will take care of the layout and presentation itself. So, here we shall begin looking at how exactly you tell latex what it needs to know about your document.**

**+ [SUBSECTION](Top Matter)**

**+ The first thing you normally have is a title of the document, as well as information about the author and date of publication. In latex terms, this is all generally referred to as top matter.**

**+ [SUBSECTION](Author Information)**

**+ It is common to not only include the author name, but to insert new lines after and add things such as address and email details. For a slightly more logical approach, use the AMS article class and you have the following extra commands:**

**+ [ITEMIZE]**

**+ [ITEM]**

**+ The author's address. Use the new line command**

**+ [ITEM]**

**+ Where you put any acknowledgments.**

**+ [ITEM]**

**+ The author's email address.**

**+ [ITEM]**

**+ The URL for the author's web page.**

**+ The first thing you normally have is a title of the document, as well as information about the author and date of publication. In latex terms, this is all generally referred to as top matter.**

由于在HTML中,section和subsection层次相同,因此在AST中,section和subsection也可以归在同样的层次中.

原先给出的生成AST的算法中,sections使用了递归,导致树的结构不够清晰.因此我进行了修改,使得所有的section和subsection都在同一层次

将Latex转为HTML的函数:

**def Node2Html(node: Node) -> str:**

**"""**

**将Node对象转换为HTML格式的字符串。**

**Args:**

**node: Node对象。用来表示Latex文档的AST**

**Returns:**

**HTML格式的字符串。**

**"""**

转换title;

**f"""<div class="title">{section\_node.getChildren()[0].getData()}</div>\n"""**

转换AUTHOR:

**f"""<div class="author">{section\_node.getChildren()[0].getData()}</div>\n"""**

转换section:

**<div class="section">  
 <h2>{current\_section\_title}</h2>  
 {current\_section\_content}  
</div>**

转换subsection:

**<div class="subsection">  
 <h3>{current\_section\_title}</h3>  
 {current\_section\_content}  
</div>**

转换ITEM和ITEMIZE:

**{items\_html}  
<li>{list\_item.getChildren()[0].getData()}</li>**

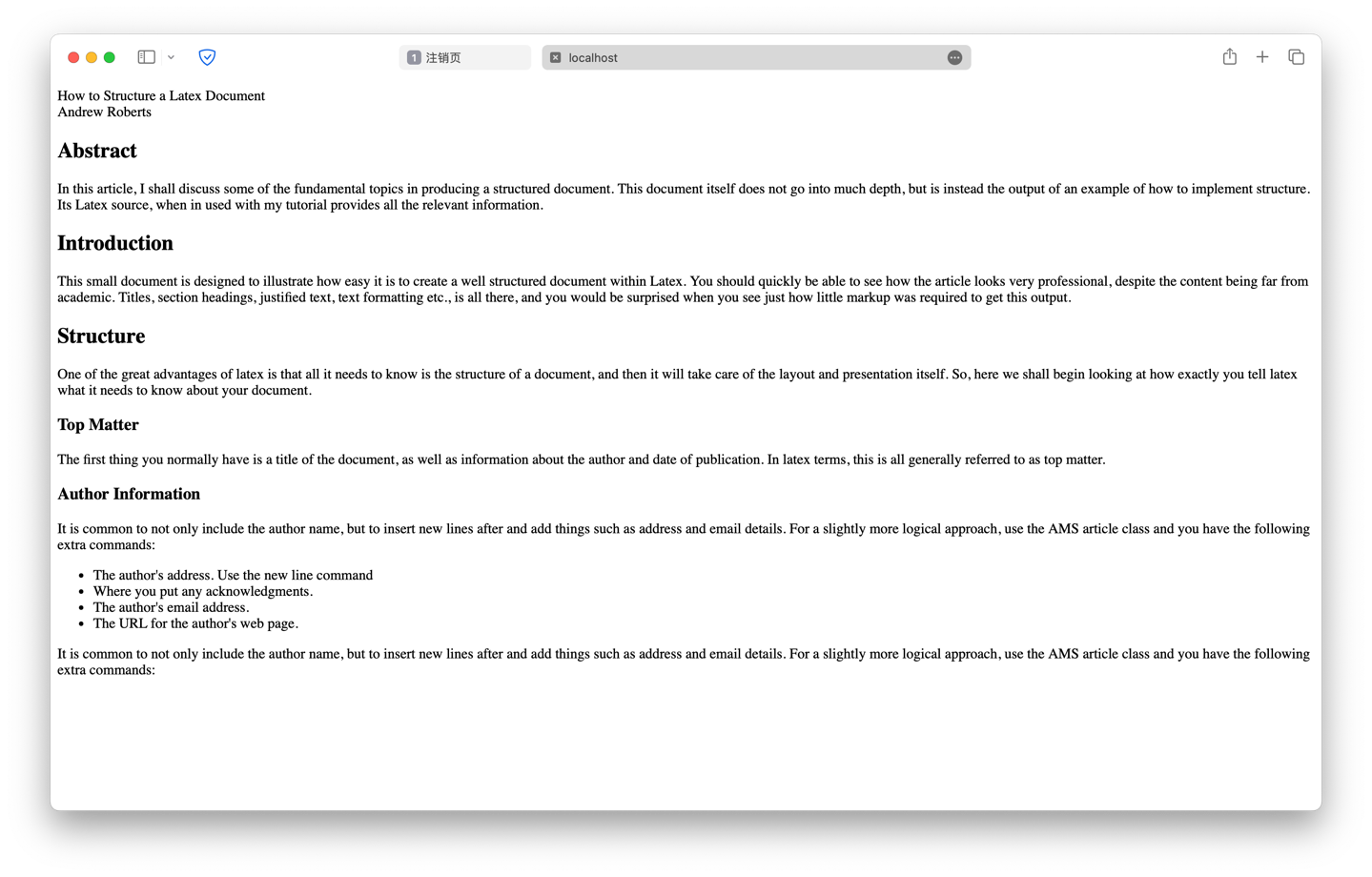
**{current\_section\_content}  
<ul>  
 {items\_html}  
</ul>**

example2.tex转换后的HTML:

**<!DOCTYPE html>  
<html>  
<body>  
<div class="title">How to Structure a Latex Document</div>  
<div class="author">Andrew Roberts</div>  
  
 <div class="abstract">  
 <h2>Abstract</h2>  
 <p>  
 In this article, I shall discuss some of the fundamental topics in producing a structured document. This document itself does not go into much depth, but is instead the output of an example of how to implement structure. Its Latex source, when in used with my tutorial provides all the relevant information.   
 </p>  
 </div>  
  
 <div class="section">  
 <h2>Introduction</h2>  
   
   
 <p>  
 This small document is designed to illustrate how easy it is to create a well structured document within Latex. You should quickly be able to see how the article looks very professional, despite the content being far from academic. Titles, section headings, justified text, text formatting etc., is all there, and you would be surprised when you see just how little markup was required to get this output.   
 </p>  
   
 </div>   
   
 <div class="section">  
 <h2>Structure</h2>  
   
   
 <p>  
 One of the great advantages of latex is that all it needs to know is the structure of a document, and then it will take care of the layout and presentation itself. So, here we shall begin looking at how exactly you tell latex what it needs to know about your document.   
 </p>  
   
 </div>   
   
 <div class="subsection">  
 <h3>Top Matter</h3>  
   
   
 <p>  
 The first thing you normally have is a title of the document, as well as information about the author and date of publication. In latex terms, this is all generally referred to as top matter.   
 </p>  
   
 </div>  
   
 <div class="subsection">  
 <h3>Author Information</h3>  
   
   
   
   
 <p>  
 It is common to not only include the author name, but to insert new lines after and add things such as address and email details. For a slightly more logical approach, use the AMS article class and you have the following extra commands:   
 </p>  
   
 <ul>  
   
   
   
   
   
 <li>The author's address. Use the new line command </li>  
 <li>Where you put any acknowledgments. </li>  
 <li>The author's email address. </li>  
 <li>The URL for the author's web page. </li>  
 </ul>  
   
 <p>  
 It is common to not only include the author name, but to insert new lines after and add things such as address and email details. For a slightly more logical approach, use the AMS article class and you have the following extra commands:   
 </p>  
   
 </div>  
 </body>  
</html>**

可以发现example2.tex中的标签均成功匹配

在浏览器中查看example2.html



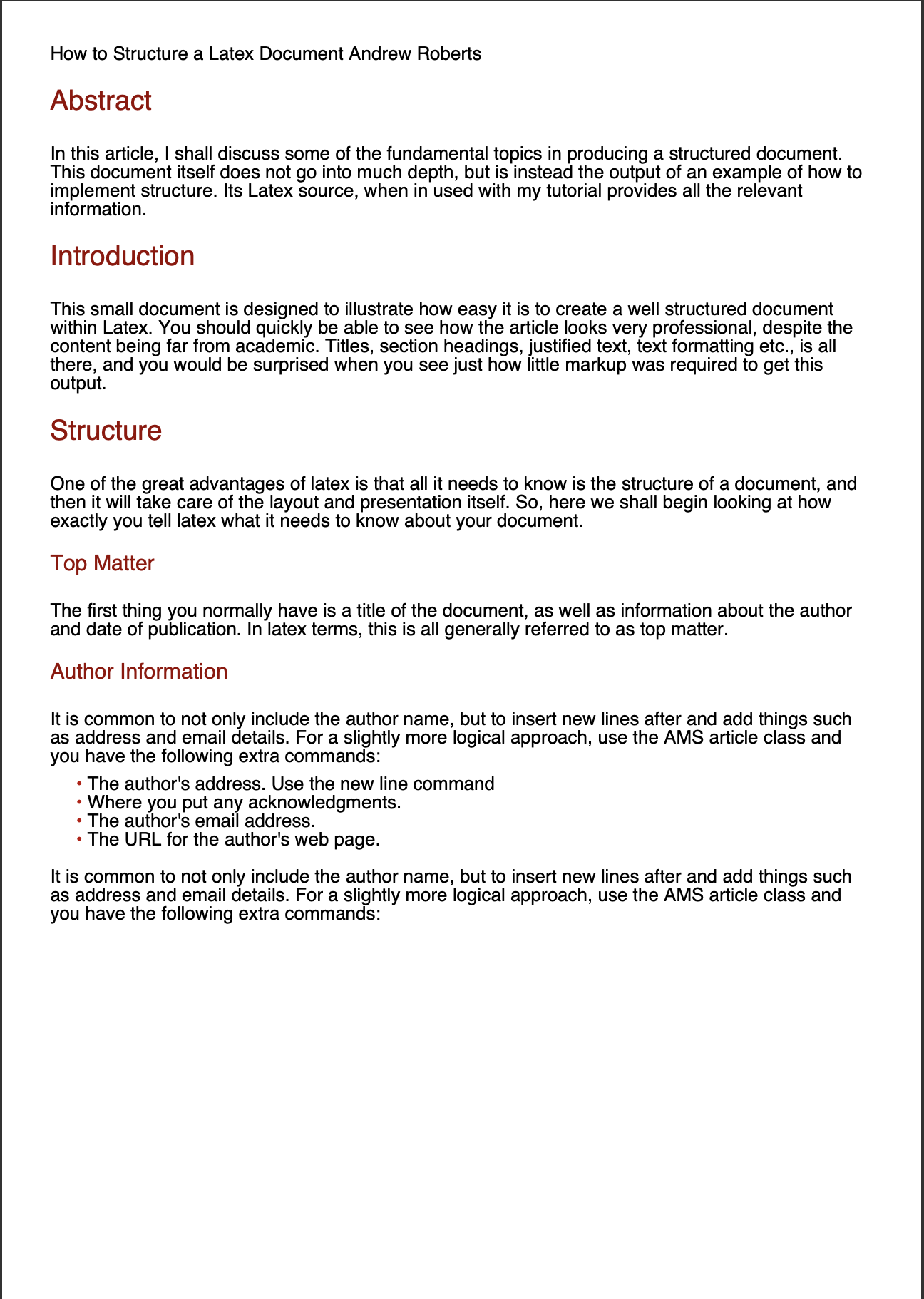
发现Latex文件中的各种标签均被匹配,生成的HTML文档格式正确

最后使用FPDF来实现将HTML转换为PDF文件

用来实现转换的函数:

**def Converter(tex\_filename: str, pdf\_output\_filename: str, html\_output\_filename: str = "") -> None:  
 """  
 将tex文件转换为pdf文件  
 Args:  
 tex\_filename: tex文件的路径  
 pdf\_output\_filename: 输出的pdf文件的路径  
 html\_output\_filename: 输出的html文件的路径  
  
 Returns:  
 None  
 """**

最终生成的example2.pdf



发现Latex文档已经被成功转换为了PDF

1. 实验总结

通过本次实验，我加深了对LaTeX文档结构的理解，学会了如何使用Python来解析和转换文件格式。遇到的挑战包括解决依赖项冲突、调试AST转换中的错误，以及优化HTML到PDF的布局转换。实验过程中，我学会了更高效地调试代码，同时也认识到了代码的模块化和文档编写的重要性。对于未来的工作，我计划改进错误处理机制，并提高转换工具的用户友好性和健壮性。