**PyAutoHomework题面**

PyAutoHomework是一个用于清华大学网络学堂的自动批阅docx作业的程序，支持多种题目类型，可生成供网络学堂批量上传的阅卷分数与批阅细节。

PyAutoHomework需要批阅者提供一个和本文档类似的answer.docx模板，一个题目元数据settings.json，一个可选的供网络学堂批量上传的template.xls模板，每个学生需提交一个和answer.docx几乎一模一样的docx。answer.docx模板中需要用户作答的区域使用特殊的task符号标出，我们通过加权的序列对齐算法提取出与用户提交的docx中与answer.docx中对应位置的字符串/图片，依照settings.json所设置的判卷规则进行判卷。例如假设这里有一道填空题：1+1= $:>ExampleBlankFillingQuestion1|2|1.0<:$ 。这里设置了一个TaskID为“ExampleBlankFillingQuestion1”的任务，标准答案为字符串“2”，这个空占1.0分。一个任务的作答区域用“<:$”符号与“$:>”括起来，中间包含若干个“|”分隔符分割参数，在下发给学生的docx中，下划线部分应为空（可以参考data文件夹中的几个用户的例子）。

**一、填空题例子：**

直接在原文中填空：1+2= $:>1\_1|3|1.0<:$ ，2\*2= $:>1\_2|4|1.0<:$ 。

更推荐的方法是将答案填入表格中（程序能更准确的匹配到结果）

|  |  |  |
| --- | --- | --- |
| **Question** | 3+3=? | 4+4=? |
| **Answer** | $:>1\_3|6|1.0<:$ | $:>1\_4|8|1.0<:$ |

**二、选择题例子：**

2.1直接在原文中选择：1+1= $:>2\_1|B|1.0<:$

1. 0
2. 1
3. 2
4. 3

2.2更推荐在表格中选择：2+3等于几？

1. 0
2. 5

|  |  |
| --- | --- |
| **Question** | 2.2 |
| **Answer** | $:>2\_2|B|1.0<:$ |

1. **手动判题的例子**

对于比较复杂的主观题，或者画图题，可以设置题目类型为“人工”。在任何需要人工判题的区域中，所有的文字会显示到终端，所有的图片会显示到屏幕上，阅卷人需要手动输入这道题的得分。

例如：请简要介绍人工智能是什么？

$:>3\_1|请简要介绍人工智能是什么？|10.0|MANNAL<:$

（这个TaskID为3\_1的空的参考题面为“请简要介绍人工智能是什么？”，参考满分为10分，属于MANNAL（手工）类型。将需要判卷人手动输入分数）

更推荐的方法是用表格来进行手工判题：

|  |  |
| --- | --- |
| 题面 | 请使用python的matplotlib随意绘制一个折线图，并解释 |
| 你的作图 | $:>3\_2|matplotlib绘图|10.0|MANNAL<:$ |

系统对于需要手动判题的题目会缓存历史的判题结果到“mannal.json”，可以在判题结束后修改分数。

1. **含部分分的例子**

假如一道题的判题规则是结果正确直接满分，结果不对需要人工依据答题过程判分，可以使用这个例子：请问1+2+3+...+10等于多少？

|  |  |
| --- | --- |
| 答案 | $:>4\_1\_0|55|10.0|JUMP|4\_1\_1<:$ |
| 过程 | $:>4\_1\_1|4\_1的过程|10.0|MANNAL|SUB<:$ |

JUMP标签表示未达到满分时使用下一题号（例子中是4\_1\_1）的分数作为结果。

SUB标签表示这是一个JUMP任务下属的子任务，JUMP不跳转时不会被自动判分。

JUMP和SUB标签可以支持复杂的高阶递归判题或多个答案的判题：

1. **使用更复杂题目规则设置**

使用settings.json可以更灵活的设置题面的规则。对于任何一个task标识符，会被解析为(taskID|arg1|arg2|arg3....)，若arg为空，则会搜索settings.json里taskID对应的规则。

例如$4\_1\_1|4\_1的过程|10.0|MANNAL|SUB$这个任务，等价于$4\_1\_1$，且在settings.json里有如下项：

1. {
2. "tasks": [
3. {
4. "taskID": "4\_1\_1",
5. "args": [
6. "4\_1的过程",
7. "10.0",
8. "MANNAL",
9. "SUB"
10. ]
11. }
12. ]
13. }

对于更复杂类型的问题，推荐直接使用settings.json配置判题逻辑。

1. **使用正则表达式匹配填空题**

具体配置逻辑可以参考settings.json:

|  |  |
| --- | --- |
| 题面 | 填写任何一个小数即可获得满分 |
| 答案 | $:>6<:$ |

使用settings.json中加入REGEX参数表示答案字符串按正则表达式理解。正则表达式无法直接在docx中用task标志表示，因为“|”字符会被混淆而无法解析。

1. **拥有多个答案的题**

具体配置逻辑可以参考settings.json:

|  |  |
| --- | --- |
| 题面 | 英语中One的中文怎么写？ |
| 答案 | $:>7<:$ |

将以此比对答案列表中的表项，使用最靠前匹配表项的分数，若全部匹配不上则为0分。

1. **其他参数**

在文本task标记中或settings中的args域，除去上述所提及的“REGEX”、“JUMP”等参数外，有一些其他参数可以提供更高级的功能。

8.1 “NOCOMMENT”参数若出现在args中，表示该题不需要提供log，手工判卷时将跳过填写评语的环节，自动阅卷时将自动将正确与否替换为“Invisible”。

8.2 “LOWERCASE”参数出现在args中时，自动型任务会将用户的文本替换为小写字符后再进行文本比对。

**九、特殊程序阅卷**

如果题面对自由度有着很高的要求，可以通过在项目目录中建立/program/spj.py程序支持自定义阅卷逻辑。该程序需要支持def run(tasked:str)的入口函数，并且返回一个包含’score’和’log’返回值的dict对象。

|  |  |
| --- | --- |
| 题面 | 请写一个大于零的奇数在下方 |
| 答案 | $:>9||5.0|PROGRAM<:$ |

遇到PROGRAM标签意味着改题需要spj (special judge)，程序将会调用/program/spj.py子程序进行阅卷。