**工艺（方法）类发明专利技术交底书**

|  |  |
| --- | --- |
| **发**  **明**  **名**  **称** | **内容：**  一种基于哈希树算法实现的PDF文件高效自动化注释方法 |
| **注释：**1、清楚简明反应本发明的主题及类型；。  一种对指定格式PDF电子表格文档实现标准化，模块化，可迁移的高效自动化注释方法 |
| **背**  **景**  **技**  **术** | **内容：**  **基本名词解释：**  STDM（Study Data Tabulation Model）是一个规范化的数据标准，用于规范化临床研究中的数据收集、数据管理及数据报告过程。它是由CDISC（Clinical Data Interchange Standards Consortium）提出的一种规范化的数据模型，可以有效地减少数据错误、提高数据质量，并且可以使得不同的系统可以互相匹配和连接，加快临床研究数据的转化和分析效率。  CRF（Case Report Form）是一个标准的数据搜集表格或电子表格，用于收集临床试验中的数据。它是一种结构化的数据搜集方法，用于收集试验过程中的各种数据和信息。CRF定义了试验所需要的数据，包括治疗及随访活动、基线和随访数据、临床结果、实验室检查、安全等信息。  Annotated Case Report Form（aCRF）是CRF的一种变体，主要用于临床试验的数据管理和分析。相比于传统CRF，aCRF除了基本数据收集外，还包括了详细的基于STDM的标记和注释。这些标记和注释加强了数据的准确性、一致性、可理解性和可审核性。aCRF可以帮助试验人员、审核人员和数据分析人员更好地理解收集到的数据，从而避免因为CRF中的输入错误、不规范、不完整、不清晰等问题所产生的数据质量问题。aCRF可以通过标记和注释来保证数据的完整性和精准性，并缩短数据分析和报告的时间，提高研究效率。  在整个临床研究中，STDM、CRF和aCRF都是至关重要的，它们确保了试验数据的准确性和一致性。STDM为临床试验提供了一个标准模型，有助于研究人员更有效地收集和管理数据。CRF是数据的核心收集工具，确保数据的质量和完整性。而aCRF则可以通过标记和注释来根据STDM提高CRF的质量，缩短数据分析和报告的时间，提高研究效率，同时也保证了数据的准确性和完整性。因此，在临床研究中，STDM、CRF和aCRF的作用是不可替代的，只有通过它们的有效使用，才能获得高质量、可靠的研究数据，并为临床医学的发展提供支持。  **aCRF的具体解释与举例：**  aCRF是用于收集试验数据的CRF的变体，与普通CRF不同的是，它包括了更详细的数据标记和注释。这些标记和注释加强了数据的准确性、一致性、可理解性和可审核性，确保数据收集的质量和完整性。  aCRF的过程比较简单。研究员提供一个基础的数据收集表格（CRF），然后系统通过自动化工具添加标记和注释来完善数据，此外，aCRF还可以通过人工审核来进一步完善数据。这让数据变得更加精确可靠，提高了数据分析效率和分析结果的可靠性。  在临床研究中，数据收集和管理是至关重要的部分，aCRF可以做到减少数据错误、提高数据质量、缩短数据分析的时间、快速地检出数据异常等。由于aCRF可以提高数据的质量和准确性，并缩短数据分析和报告的时间，所以在整个数据分析过程中，aCRF是不可或缺的一环。  aCRF在整个数据分析链条上的速度制约因素也非常重要。缺乏准确、可理解和标准的数据会导致数据分析师花费更多的时间来分析和解释数据。而aCRF通过自动化标记和注释等工具，可以缩短分析师分析数据的时间，提高分析效率，并且提供可靠和准确的数据结果。  因此，在临床研究中，aCRF的实际效果和应用影响整个临床研究的进程和流程，提高了数据质量和准确性，加速了分析过程的速度和过程，相应地，根据分析结果进行对临床前景的评估也更加准确和可靠。 |
| **注释：**简述现有工艺（方法）或方法存在的问题和缺点。 |
| **发**  **明**  **目**  **的** | 内容：  本发明的目的是对任意PDF版本的CRF表格，实现自动化注释，并实现版本迁移和模块化分割和合并操作。  aCRF是一项非常耗费人工的工作，需要一个工作人员去编写和校对。目前针对这一过程，出现了一系列软件和分析流程，下面简述如下：   1. 从制表前基于源代码进行注释添加   这些操作都是在生成 CRF的PDF文件之前对CRF的原始表格进行分析转换，并把相关的注释结果输出给相应的PDF制作工具和CRF的原始信息一起输出成PDF文件   * 1. 基于SAS的Mock Shell添加   SAS的 Mock Shell是一种用于标记和注释CRF数据的自动化工具。Mock Shells工具可以自动分析CRF中的数据结构，并为其生成数据标记和注释。这些标记和注释具有相当好的可读性，可以为数据管理人员和分析人员提供更好的数据。该方法是在制表前，把所有的相关标题或者内容文字输出给SAS软件中，并使用Mock Shells 按照 表—>标题名—>内容的形式映射到SAS的数据库中进行查询并输出注释，最后添加到指定位置   * 1. 基于LaTeX添加   LaTeX是一种功能强大的排版工具，广泛用于学术界和出版业。部分CRF最终的PDF版本是由LaTex排版文件直接生成的。可以自己写代码或者使用相关工具直接对Latex原始文件进行编辑，并在指定位置输出注释文本框。这需要制表人员对真个表格的结构烂熟于胸，知道哪个位置具体的文字所代表的意义和其附近是否有个空白空间可以添加注释。   * 1. 数据库直接添加   部分CRF的PDF文件基于 Oracle等成熟的数据库系统直接导出，在其原始数据库底层表格中，添加注释信息和所在坐标，可以直接生成PDF。类似的工具有Oracle®Clinical（Oracle，美国），Clinsys®（Jubilant Organosys，美国），锐度测量® （Meditata解决方案）RedCap等   1. 制表后添加   大多数情况下，做注释和做表格的不是一个团队。注释人员只能根据没有注释的空白 PDF上进行后期注释，目前公布了以下一些方法：   * 1. 基于制表说明文件 Study Design Specification (SDS) 进行注释添加   制表团队在输出空白PDF的同时，可以选择性的同时输出一个Excel表格，该表格记录了PDF文件每一页的内容，所在页码，数据类型和坐标，该文件被称为SDS文件。在此基础上，注释人员根据这个表格的内容自行添加指定的列，比如注释的颜色，注释的内容，注释文本框的坐标偏移量等信息。而后根据这一表格的信息使用相关工具为空白的PDF添加注释。    Figure 1 SDS文件与注释结果PDF  其中SDS中的D列和E列是后添加的信息，不是SDS原始信息   * 1. 基于表格文本匹配进行添加      1. 特征文本匹配法   这一方法提前准备一个表格文件，里面写上一些表头包含的单词和其对应得的注释信息。而后单独提取PDF文本信息，并按照换行符或者空格进行断句分割。该方法要求代表表名或者表头的字符具有一定特征，比如被括号包含起来，或者被\*包围起来，并以此判断扫描的是拿一张CRF表格的位置，其余文本被作为关键字进行保留。检查关键字的内容是否出现在表格对应的文件中，如果是，则把注释信息放在这个单词旁边。比如Figure 2的例子表名和表头都用\*进行了修饰，几个星号就是几级标题。让程序对文本进行解析，根据特征匹配得到标题级别和页码，标题级别和标题名进行逐级匹配并将 国男、国女、外男、外女四个注释放到对应的位置下。    Figure 2基于文本匹配的注释方法   * + 1. 文本内容编号法   另一种方法是逐页、逐行扫描页面中的每一个单词捕获其中的内容和坐标，其中每一行的内容都进行编号，并记录在一个Excel中，而后编辑这个Excel并写代码还原注释。    Figure 3基于预编号系统进行注释   * 1. 基于BookMarks进行添加   让员工提前看一遍这个文档，并在文档的每一页打上一个书签，书签的内容是这一页对应的表格名和其内部出现得而表头名。而后写程序，根据这些书签将对应的表头的注释信息以文本框的形式添加到该页面的一个角落中，而后打开PDF编辑器，手工将这些包含注释的文本框信息拖拽到对应的文本旁边。   * 1. 基于XFDF文件进行编辑   XFDF文件是一种XML文件格式，​用于将数据直接插​入到PDF中。​ XFDF文件可以用于将多个表单中的​用户信息填充到PDF文件中。​XFDF文件可以使用PDF阅读器打开，​如Adobe​ Acrobat Reader​等。任务开始时，由人工创建一个XFDF文件，手动将文本框的坐标输入到XFDF文件中，并在XFDF文件中书写注释信息。保存后，将XFDF文件和空白PDF同时加载到编辑器中，并另存为最终的注释结果。  **存在的不足：**   1. SDS等文件需要表格生成方提供。这些表格生成方一般是国外公司，比如Oracle公司等，其在国外沟通起来困难。并且，除非公司领导出马，注释团队基本上拿不到SDS文件。 2. 在数据填写时，可能有些表格内容要延长，导致最终文档的页码和SDS文件页码对不上。 3. 表格是逐步分发并最终整理的，所有表格的页码可能是乱的，和SDS文件对不上。 4. 如果使用预编号系统，由于编号和注释都是和页码对应的，如果文档页面顺序发生变化则该方法失效。并且，在进行检查的时候，由于所有内容的组织结构被编号打乱，难以进行检查。比如如上例子中：1(页码)🡪**国籍样本调查表**🡪**中国**🡪**性别**🡪**男** 添加注释为**国男** ，而 1(页码)🡪**国籍样本调查表**🡪**其他**🡪**性别**🡪**男** 应该被注释为**外男。**放到预编号系统里则变成了1（页码）🡪4🡪男 添加注释为**国男**，而1（页码）🡪8🡪男 添加注释为**外男**————这种信息非人类可读，难以进行后期检查。 5. 对于特征文本匹配法，其要求表格的表头和标题级别和内容都使用特征文本进行文本标注，比如#或者\*号或者括号。如果采用颜色、字体、字号、字体、文字边距、特殊标注等等其他方法对标题级别进行区别，用于该方法一上来就把所有内容变成纯文本，这些特征都丢失，无法匹配。而且该方法使用R语言的tm package实现，该packge有bug，一旦文档中出现图片，后面解析得到的文本都是乱码。 6. 采用特征文本匹配法的话，需要提前知道标题最多有几个级别。大部分的预编译语言如C，Golang，Java，Javascript等都需要指定字典的维度并预先构建结构体用于数据存储，比如Golang和Java，装一级标题的注释需要构建一维度hash表，二级标题构建二维度哈希表结构体，Hash[一级标题][二级别标题]=注释结果 这种数据结构，三级标题需要构建Hash[一级标题][二级标题][三级标题]=注释结果 这种结构体，四级标题，五级标题以此类推。其代码根本无法书写。目前常用的方法是通读文档，知道最多有几个级别的标题，而后在代码的声明中预先构建多个数据结构并进行交叉比对——这么做费时费力，代码维护极其困难，会占用海量内存，并且一个CRF PDF文件只能和一个程序对应，无法做到通用性。 7. 文档是PDF文件，其展示结果和其编码可能不一致，比如某些标题文字为了展示美观内部会插入一些空格和制表符，使用SDS，目前的方法都只支持文本精确匹配。哪怕多了一个句号都无法匹配。这也会在项目中出错。 8. 注释PDF不是一锤子买卖，在项目进行中，PDF文档结构，注释的内容可能要有多个版本更迭。每一次更迭都需要修改注释的一部分，如何做到注释迁移也是一个大问题。以上方法都和坐标还有页码绑定，一旦注释或者PDF文档结构坐标发生变化，所有工作都要推到重来，简直就是噩梦。 9. 每个CRF都是至少3年多期临床研究的资料汇总，其数据量庞大，一次CRF的PDF文档至少200页以上，多的可能到万页。一个人做会带来超大的工作压力，迫切需要一种能够分割合作的工作和编码机制。目前所有的注释方法都依赖于页码还有预设坐标体系。因此，只能采用文档分割方法，即把一个CRF的PDF文件按照表格分成多份并保留每一页的页码，把不同部分分给不同的人去构建注释表格或者修改SDS文件。项目要求每个人严格遵守页码，和文档内容的坐标，即便内容超出，也要想方设法的不能分页，因为一旦分页，页面和坐标都会发生变化，别人的工作就无效了。最后由管理员把所有的SDS或者预设表格文件合并起来。进行注释。 10. 对于没有SDS文件的项目，全公司就指着一个员工先构建好BookMarks或者为每一页内容编好码，然后全公司去做注释，这简直就是噩梦。现在又赶上新冠流行，多次出现一人病倒，全公司乱套的现象，尤其是存在国际团队的跨国公司，这种现象简直就是此起彼伏。   针对以上这些问题，我们迫切需要一种注释方法，要求有以下优点：   1. 放弃SDS数据等一切前置数据。由于SDS文件基本上拿不到，所以需要新方法能够根据填写好的文档自动化生成SDS文件或者类似结构文件。或者干脆有自推断算法计算PDF表格文档的结构。放置BookMarks或者文本内容提前编号，这两项内容花费大量人力是天怒人怨的工作，简直就是绝交利器。基于纯文本匹配的算法需要表格设计人员抓耳挠腮的设计各种特殊字符为不同级别的标题做标注，也十分困难。而且还要尽可能避开图片——也就是说各种公司LOGO也决不能出现，这简直无法接受。 2. 放弃页码和绝对坐标体系，转而使用相对坐标体系。我们对于某个文本坐落于哪页哪行的坐标完全不感兴趣，我们要求是在他旁边左右或者上下放下一个注释文本框。所以一个写死了这个文字在PDF的某一页某个坐标的数据对我们来讲完全没意义，而且还会形成桎梏—— 一旦文档结构或者页码顺序发生变化，他会让我们的一切努力化作乌有。我们需要的是个能够识别到指定文档中指定表格指定标题级别的指定内容的坐标，并在他旁边放一个注释文本框，其宽度和高度能够根据注释内容自动调整。并且注意，某一个标题本身的内容没有意义，因为其在文档中可能出现多次，其上下游多级标题的组合形式才是绝对唯一的，如Figure 3中的**性别**+**男**or**女**这种标题组合都不唯一，需要 **国籍样本调查表**🡪**其他**🡪**性别**🡪**男** 这四层标题组合才能确定其唯一性。 3. 从编程角度讲，需要存储这些数据的数据结构具有自拓展性，即需要递归的让某一个对象的某一个子方法的数据结构还是自身。这在内存布局上具有不确定性。所有的预编译语言，就是申请个对象或者变量需要New Make Void Assert public or Privacy这种把变量的结构形式或者大小作用域订死得而语言全部淘汰。我们需要赋值有弹性甚至是内存垃圾回收机制存在懒惰性的语言。 4. 在支持字符串严格对应的前提下支持模糊搜索，尤其要容忍大小写错误和空格等非数字字母，防止由于展示效果美观而添加的空格等占位符、大小写书写等问题。如果这些字符串（即标题或者表名）没法完美匹配，则使用正则匹配进行搜索。 |
| **注释：**针对现有技术的不足，本发明专利的优点是什么？或解决了什么问题？ |
| **发**  **明**  **内**  **容** | 内容：本发明设计并实现了一种基于递归嵌套的无限维哈希树算法的数据结构，将其用于存储PDF表格数据的解析结果。其原理是，构建一个包含表格一切内容以及其上下游组织关系的超级目录并将其保存成文件。工作人员编辑这个超级目录文件，并在需要标注的地方上插入字符串即可实现标注。而后，实用程序对照原始PDF文件和这个修改过后的超级目录文件实现自动注释。  这么做有以下优点：    Figure 4．哈希树举例  该方法使用的理论基础——哈希算法是理论上最快的查询算法，效率最高。  哈希树并不记录每个字符元素的坐标和出现页码，甚至连顺序也不记录，他只记录字符串或者不同级别标题之间的隶属关系，比如哈希树记录得是如上图Figure 4所示，可以发现这个哈希树中记录的只是 表2 表1 不同级别标题的隶属关系，标题的顺序都是随机的，在哈希结果中完全没有意义，并且，表1 和表2两个子表之间内容完全独立，互不影响。这样就可以实现任务的无损分割和分发与并行。  同时，哈希树中只保留了PDF文档的组织结构。在流程的最后一步，程序还会对整个PDF文件进行遍历，只有碰到需要注释的出于特定文档结构上的特定文本元素时，才会调取该元素在文档中的绝对坐标，并利用这一坐标数据进行计算实现在其旁边放上注释文本框的操作。这意味着即便文档结构，甚至相关元素的绝对坐标发生了变化，由于我们流程的最后一步是扫描这个文档并获取坐标，因此会跟着发生变化，之前的坐标啊，页码顺序啊对我们不会有任何影响。    Figure 5 注释迁移模型举例  **结果的高度可迁移性。**Figure 5描述的是一次注释迁移过程。子图（A）为老版本文档经过注释后得到一个记录文件，（B）为新版本的文档的目录结构文件，可以发现新版本里删掉了一个表——“表3” 基础知识：哈希表与无限维哈希树： 哈希表（Hash Table），也叫散列表，是根据关键码值（Key-Value）进行直接访问的数据结构。它通过把关键码值映射到表中一个位置来访问记录，以加快查找的速度。这个映射函数叫做哈希函数（Hash Function），存放记录的数组叫做哈希表（也叫散列表），工作时一般需要开辟一块较大的地址空间作为哈希表。哈希表的主要优点是查找效率高，能够在时间复杂度为O(1)的时间内找到所需元素，对于大数据量的搜索或排序任务，哈希表的效率明显优于其他数据结构。  哈希树（Hash Tree）也称为梅克尔树（Merkle Tree），是一种树状数据结构，它是一种用于验证和管理数据的有效方式，通俗地说就是一个可以存储多维数据的哈希表。当数据被写入哈希树时，每个节点都用它的哈希值来代表整个子树的数据。当树中的数据发生更改时，只需要修改这个特定节点和与它相关的哈希值，因为其他节点不会受到影响。  在无限维哈希树中，每个元素都有一个唯一的键值。这个键值可以用来计算出在哈希表中的位置。计算这个位置通常使用哈希函数，这个哈希函数通常只考虑一个维度的键值，并返回一个具体的哈希值。在每个维度上，都有一个对应的哈希函数来计算在该维度上的哈希值。  无限维哈希树的主要优点在于其灵活性和可扩展性。它可以动态地增加或删除维度，使其非常适合存储多层嵌套的数据。因此，无限维哈希树在很多应用场景下都得到广泛应用，例如机器学习、推荐系统、图像处理等等。  无限维哈希树在数据查询和组织上具有以下优点：   快速查询：无限维哈希树使用哈希函数来计算每个元素在哈希表中的位置，这使得查找操作非常快速，平均时间复杂度为O(1)。对于大数据量的搜索或排序任务，无限维哈希树的效率远高于传统数据结构。例如，如果我们将一个学生名单存储到哈希表中，可以通过学生ID快速查找每位学生的信息。并通过诸如体重，身高，性别等第二级别的键快速得到某个学生的某项信息。理论上讲一个无限维哈希树可以存储任何一个表格的全部信息和组织结构而不用考虑信息出现顺序   数据组织：无限维度哈希树可用于存储多维数据，这使得它在数据组织方面非常有用。它可以用于存储具有多个属性的对象，其中每个属性可以看作是一个维度的键值。这使得数据管理和访问更加灵活和易于组织。   动态添加或删除维度：无限维哈希树可以根据需要动态地增加或删除维度，这赋予了它超出传统哈希表的灵活性。这意味着它可以处理多维数据，而无需提前固定多维度，这在某些应用场景中非常有用。   低内存消耗：无限维哈希树在存储时可以高效地使用内存，具有较低的内存消耗。这特别适用于处理大量数据的场景，能够有效减少内存使用，并提高处理性能。   易于维护：因哈希表通常使用简单的键值对数据结构，因此维护非常简单，尤其是数据量较小的时候。如果我们需要查找一个用户的一些数据，只需使用其唯一的ID就能轻松检索到该用户的信息。与传统的关系型数据库相比，使用哈希表更加自然，并且维护成本较低。  总之，无限维哈希树在多维数据存储、快速数据查询以及灵活性方面都提供了优秀的解决方案，适合应用于处理大规模的数据。 任意表格的可哈希性： 根据数学推导，任意表格都可以被转换为有限维度的哈希树。数据表格转换为哈希树的推导思路如下：  数据表格可以看作是一个二维矩阵，其中的每个行和列都可以看作是一个维度。  哈希表是一种多维数据结构，可以将每个维度看作是键值，将所有维度的键值组合在一起就可以唯一地确定哈希表中的一个位置。  因此，我们可以将数据表格转换为一个有限维度的哈希表即哈希树。具体转换方式是选择一个或多个维度，将这些维度的键值作为哈希表的键值，将行或列的数据作为哈希表的值。 哈希树与JSON数据格式： JSON是一种轻量级的数据交换格式，常用于将数据从服务器传递到客户端。它是一种纯文本格式，易于阅读和编写，可用于各种编程语言之间的数据交换。JSON由键值对组成，其中键必须是字符串，而值可以是任何有效的JSON数据类型，如数字、字符串、布尔值、数组或对象。  在实际应用中，哈希树和JSON也可以相互转换，例如将哈希树中的数据转换为JSON格式以便进行网络传输或存储，或者将JSON格式的数据转换为哈希树以便进行快速的数据检索和处理。此外，Json由许多专用的编辑器，可以很方便的展示给用户并进行数据编辑。 多个无限维哈希树递归更新 多维哈希树的Update函数可以用于更新哈希树中已有的元素。对于一个n维哈希树，Update函数通常采用递归的方式实现。其基本流程如下：   * 1. 检查给定的键值是否存在于哈希表中。   2. 如果不存在，可以选择将元素插入到哈希表中，也可以忽略此操作。   3. 如果存在，那么检查当前哈希表节点（或子哈希表）的维度数。   4. 如果当前哈希表节点的维度数等于1，表示该节点对应的是一个单一的数据元素，直接更新该元素的值。   5. 如果当前哈希表节点的维度数大于1，表示该节点对应的是一个子哈希表，需要递归地调用Update函数，继续检查下一个维度的键值，直到处理完所有维度。   6. 如果仍然无法找到对应的元素，可以选择将该元素插入到哈希表中，也可以忽略此操作。   7. 根据需要，可以更新整个哈希表或者只更新指定元素的值。   8. 需要注意的是，为了防止哈希表出现过多散列冲突，Update函数需要根据设计原则合理选择哈希函数，并及时进行哈希表的扩容。此外，Update函数还需要考虑多线程并发更新的问题，通常采用锁或原子操作来保证数据的一致性和线程安全性。   由于递归Update函数的存在，两个多维哈希之间可以实现以一个哈希为准，快速更新另一个哈希的值 发明内容无限维哈希树的算法原理和实现： 我们使用Python实现无限维度哈希。Python自带哈希结构表的数据结构，我们构建一个对象，让该对象无线嵌套自身，并部分修改其内存垃圾回收机制。没用到的哈希在代码结束前不被回收。实现了无限维度哈希。  同时，针对搜索的方法进行了重构，让其key值先进行字符串匹配搜索，如果搜索不到并且key值的list里包含正则对象（re.Pattern），则针对正则对象进行匹配并返回第一个匹配上的正则表达式所对应的哈希表对象。得到了如下效果：   1. 任何时候声明一个任意维度哈希表都无需提前声明数据结构 2. 如果声明的一个任意维度的哈希表其value不存在，则其值被定义为一维度哈希。 3. 每个维度的哈希表的值可以是任意对象包括另外一个哈希表，但只有包含字符串的数据会被 “=”取回。比如，如下代码(以下加下划线的字符为Python代码):   首先给多维哈希赋值***MultiHash[ “T1” ][“T2”]=”Comment”***  如果代码出现***MultiHash[ “T1” ][“T2”]***会返回一个空的哈希 *{}*；  如果代码出现***MultiHash[ “T1” ][“T2”][“T3”]*** 则***MultiHash[ “T1” ][“T2”]***的值实际为:  **[**  **[“Comment”],{**  **“T3”:**  **{}**  **}**  **]**  而任何等号取值符只能得到第一个字符串list部分的值。***x = MultiHash[ “T1” ][“T2”]* ,**则X的值为 ***[“Comment”]***  最后，我们将整个这个数据结构构建为一个对象（Class），方便后续调用。    Figure 6．支持正则匹配的无限维度哈希测试结果    Figure 7 哈希树模型与文档结构对应图 使用哈希树算法对CRF的PDF文件进行注释的整体流程概述 以下流程图描述了如何从一个空白的CRF PDF文件生成一个带有注释的CRF PDF文件。这是通过以下步骤完成的：   1. 使用python脚本GenerateJson.py将空白CRF PDF文件提取文档结构转变成一个哈希树结构并将数据转换为一个JSON文件。 2. 然后，可以手动编辑JSON文件来添加注释信息。 3. 将编辑后的JSON文件和空白CRF PDF文件传递给AddComment.py脚本，该程序读取JSON文件的数据结构并与空白CRF的哈希树进行比对，如果发现某一个子节点下出现了字符串——即注释信息，则在匹配该节点的坐标 在旁边画一个文本框放置注释信息完成注释。   最终，通过使用AddComment.py脚本，将注释添加到CRF PDF中，并生成批注后的CRF PDF文件。    Figure 8．整体分析流程图  下面分步讲解： 基于哈希树算法解析任意PDF表格 我们将PDF文件进行逐页逐行扫描，并可以根据之前获得的格式规范信息将整个文档的逻辑结构重构并导入到哈希树中。  哈希树是一种数据结构，它可以让我们通过多个键来快速访问和查找多个值。在这个场景中，我们可以将PDF表格中的不同级别标题作为哈希表的键，将每一行和每一列都转换为哈希表中的一维，通过多次嵌套多维哈希表来实现多层级别的存储。这个方法不仅可以让我们轻松地访问和处理表格数据，而且可以通过哈希表的快速查找功能来提升效率，减少人力和时间成本。    Figure 9．PDF表格文件基于无限维哈希树的数据展开  举例来说，在获取不同级别标题和文档内容的格式标准的情况下，通过对整个页面的逐行扫描可以对任意级别的标题进行捕获。比如以上页面：   1. 一级标题其字体为黑色，加粗，5号，Curel字体，并且被括号囊括、 2. 二级标题其字体为黑色，加粗，6号，TimesNewRoman字体，并且被方括号囊括，出现位置所在字块其X坐标距离边框不超过50像素。 3. 三级标题字体和字号二级标题一致，但是其出现位置至少距离左边框300像素。 4. 四级标题的以[A：]形式出现，并且其冒号后的内容和最终的注释结果一致。 5. 四级以上标题格式与四级一样，但是距离至少间隔100像素。   基于以上知识，我们可以使用逐行遍历的方式快速将整个文档的不同级别的标题快速扫描出来。并逐行以代码 Hash [T1][T2]……[Tn]的形式对无限维哈希树进行赋值。当然还可以有其他的逻辑，这取决于每个文档设置的不同。但是基于字体、字号、颜色、上下文字、文字坐标、特殊符号、等多个信息都可以作为解析参考。  将整个PDF文档遍历一遍后，我们就可以得到一个包含整个文档内容和组织结构的多维哈树对象。保存这个对象成为JSON格式并交给工作人员进行注释。 手工注释并获得最终结果 手动编辑解析得到的JSON格式文件，在需要添加注释的地方添加注释并以引号包围。注释完成后，再次使用程序逐行遍历空白的CRF的PDF文档。这次不是给多维哈希赋值，而是取出值。  如果指定的多级标题结构下，哈希树里有值，并且该值是字符串，则把字符串拿出来，并把对应级别标题所处的最右侧x轴坐标提取出来，在这个位置+一部分偏移量，画一个文本框，文本框颜色为默认，宽度随着多维哈希值的字符串宽度进行变化。而后把文本框和注释画到这个位置，就得到最后的结果。    Figure 10 编辑JSON并得到注释结果  如上图中红色的框内的内容是添加的注释。在编辑完JSON文件后，将编辑好的JSON文件读取到内存转换成哈希树，再次解析PDF，但是这次与之前一次扫描不同，这次是按照整个PDF文档的扫描顺序，从内存中的哈希树中取值，如果取道的值是字符串，则将这个字符串作为注释放到匹配文本的旁边。    Figure 11．添加注释的代码和相关注释 文档注释迁移 如果将每个子表格的表名作为哈希表的第一个维度的键，然后将子表格中的数据存储为哈希表的值，那么我们可以说每个子表格的表名是哈希树第一个维度的键。  在这种情况下，我们可以使用每个子表格的表名作为哈希表的顶级键，然后将每行和每列的标题作为哈希表中的第二级键和第三级键，最后将单元格中的数据存储为哈希表中的值。这样，我们就可以根据子表格的名称来方便地访问子表格中的数据，同时根据行标题和列标题来查找单元格，实现了高效的数据存储和查询。  哈希具有无序性，在解析好的哈希树中，表出现的顺序对于解析结果没有任何影响。对于出现变化的PDF文件比如添加部分标题，二级标题顺序变化，分页调整等的PDF数据文件来说，由于哈希树记录的是PDF文档的逻辑组织结构，如果某个表的内容没有发生变化，其对应的多维哈希值没有任何变化。所以，对于需要迁移的注释，只要其数据结构内部的键值对和新文档的结构能够对得上，就可以被轻松地替换过来，而不用在意出现的顺序和页码和坐标等问题。同时，如果旧文档的哈希表结构和新文档的发生了不一致，则会把不一致的结构添加到新哈希树中：这意味着在新文档最终遍历注释的过程中，这些冲突的老文档特有的哈希树枝杈由于和新文档对应不上，而不会被访问，不会对最总结果有任何影响。    Figure 12 文档注释迁移操作流程    Figure 13 文档注释迁移代码实现 文档分割与重组 一个CRF文档，至少200页起，目前主要靠一个工作人员从头做到尾，显然非常费力。使用哈希树，如Figure 5所示，可以将所有的不同的表分成不同的子树（即图中各个一级键之间完全独立），子树之间完全独立互不影响。因此，可以单独对每个表对应的子树切割出来单独注释而不会影响整体的注释效果。流程如下：    Figure 14文档分割与整合模型原理示意图  只要分割得当，可以按照表格把一个PDF文档分割成多块，每一块单独构建一个JSON文件记载单独的文档组织结构。而后由不同的人独立注释，最后使用Update函数将所有的注释结果合并回来。这实现了任务无损分割和完美整合 |
| **注释：**根据发明目的或解决的问题，工艺（方法）方法类发明应写明其步骤，各步骤中所需要的条件，如温度、压力范围、酸碱度、时间及其他具体要求。 |
| **发**  **明**  **效**  **果** | **内容：**  该方法有如下效果和有点：   1. **保留文档全部Meta信息，使得制表和操作过程方便**。每个文档部件的诸如字体、字号、坐标、等复合条件都可以作为标注不同层级文档标题或者表名的特征。比如我可以要求：文字边距大于2像素点，X轴坐标大于1000像素点，黑体，Times New Roman字体，6号字号，红色的文字是一级标题。由于本方法是直接对PDF二进制文件进行逐页逐行逐个文本块进行解析，这些Meta信息全部都可以作为特征用于匹配不同级别的标题，这无疑对制表方提供了大量方便。 2. **对PDF解析错误有高容忍度**。即便这些Meta信息不足以定义全部层级的文档标题，我们可以根据人类阅读习惯从左到右，从上到下的对文档部件进行逐个扫描，单纯按照X轴坐标和扫描顺序前后也可以得到基本正确的结果。即便我们解析的结果和预期发生了冲突，但是表格中一定不会出现两行完全一样的文字，基于此，我们也可以全部遍历出文档中全部潜在注释位置并生成JSON文件方便编辑。 3. **不受排版形式影响，对表格内容的坐标位置和页码完全不敏感**。在Hash表中key的顺序没有任何意义。因此，在实际中，我们只需要保证每个表格结构的完整性，表格内部每一页的顺序，还有表格间的顺序将不会对我们的最终解析结果有任何影响。 4. **处理速度接近理论极限**。Hash表的查找时间复杂度近乎O(1)，其查找期望时间接近于1/2 O(1)。是几乎最快的搜索算法。因此，理论上该方法在速度上接近理论极限。对比相关研究的分析速度，同时处理一个200也文档，对方6步，不算手动操作每部接近30s。本研究从头到尾6s搞定。 5. **该方法不依赖于任何前置条件或者前置人工操作**，使用程序遍历PDF文件构建JSON结构的数据文件，并可以基于编辑后的JSON文件自动化构建注释—这不依赖于任何诸如SDS文件，或者人工构建BookMarks等条件。真正实现了自动解析，自动注释，极大的节省了人力。并且，注释结果是根据搜索到的文本添加一个偏移量构建的文本框。他依赖于相对坐标，对页吗，文本绝对坐标完全没有任何依赖。 6. **实现了基于正则表达式的Fuzzy Searching功能**。对于一些常见的修饰符，比如空格，制表符等等可以指定正则表达式进行模糊搜索。同时实践中有些注释的值要随着匹配内容发生变化，这些使用正则表达式也可以完美实现 7. **高度模块化。**依赖于哈希表的无序性和哈希树子树之间得到完全独立性，实现了CRF PDF文档注释的模块化和分割操作能力，一个文档按照表格形式分割，并发送给多个人修改并注释，无论他怎么操作或者扩容，修改这个文档，对文档其他部分完全不产生影响。**这个目前是首创。** 8. **文档更新极其方便。**以前的所有方法，无论是基于SDS文件，还是文字匹配，内容编号，加BookMarks还是创建XFDF文件，都要不依赖于页码和绝对坐标，要不依赖于特定的文字排版规则。这俩体系一旦出现变动，整个前置工作全都需要做一遍，比如某一个表格的两个章节顺序发生调换、多个个表格之间的顺序发生调换，某个表格的某个留白变大或者变小，这将导致坐标和页码体系整个调整，导致依赖其的前置工作全部破灭。基于文字匹配的方法，如果文档中碰到图片，或者遇到如UTF-16，GBK36等少见的文字编码规则所有后续文字全变成乱码，整个方法崩溃。我们这个方法使用类似目录的方法构建哈希树将文档结构进行记录，完全脱离于绝对坐标体系和页码。同时，使用代码对二进制文件的文本块进行扫描，完全规避开图片等区域，不会有任何影响。同时在更新的时候，将两个不同版本文档的组织结构进行对比，把老版本的注释直接覆盖新的空白版本的哈希树：如果文档某个部分新老结构一致，老的注释直接覆盖新的空白键值对实现文档注释迁移。如果不一致，老的新的子树取并集。由于最后是对新版本空白文档进行搜索遍历，老的特有的文档结构根本不会被遍历到，导致其对新文档完全没有任何影响。 9. **数据保存极其方便。**JSON格式本身就是数据库软件数据保存的一种格式，被一系列NoSQL软件广泛接受。哈希树和JSON格式的互相装换也有非常成熟的一些列标准和工具。使用哈希树保存文档结构和注释信息并转换成JSON格式后，可以非常方便的保存到任何NoSQL数据库中。 10. **操作便捷性。**本方法不依赖任何前置数据，使用PDF文件本身构建哈希树形式的类似于目录结构的文档结构关系网，免去了不同团队互作去的前置数据的步骤。同时，由于有很多方便的JSON编辑器软件的存在，可以很好地展示整个文档的数据结构，操作人员看JSON文件就可以很容易的推导出文档结构并进行注释，这相对于其他方法，极大的简化了整个操作过程。并且，由于整个文档的结构被完整的保留下来，其对应的注释一定会出现在对应文本的旁边。极大的方便了用户对结果准确性的检查——检查对应文本旁边的文本框内容就知道是不是注释对了，使用效果极其方便。     Figure 15方便的JSON编辑器  需要加什么注释在双引号里写就好了。 |
| **注释：**本发明与现有技术相比所具有的优点和效果，最好有适当的理论和实验数据作为支持。 |
| **附**  **图**  **及**  **说**  **明** | **内容：** |
| 注释：工艺方法类发明一般提供：工艺（方法）流程图（用文字足以清楚、完整地描述其技术方案的，可以没有附图，也可以结合附图进行说明） |
| **实**  **施**  **方**  **式** | **内容：** |
| **注释：**实施方式是对技术方案的进一步描述、细化和解释；  对于方法工艺类发明，应给出详细的步骤及工艺条件（工序、压力、温度、时间等）。 |
| **其**  **他** | 其他有利于我们理解技术方案的信息。 |