(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 112262390 B
(45)授权公告日 2025.06.27

(21)申请号 201980035772.X

(22)申请日 2019.06.12

(65)同一申请的已公布的文献号
申请公布号 CN 112262390 A

(43)申请公布日 2021.01.22

(30)优先权数据
62/684,498 2018.06.13 US
62/749,001 2018.10.22 US
16/438,325 2019.06.11 US

(85)PCT国际申请进入国家阶段日
2020.11.27

(86)PCT国际申请的申请数据
PCT/US2019/036829 2019.06.12

(87)PCT国际申请的公布数据
WO2019/241425 EN 2019.12.19

(73)专利权人 甲骨文国际公司
地址 美国加利福尼亚

(72)发明人 M·马拉克 L·E·李瓦斯
M·L·克莱德尔

(74)专利代理机构 中国贸促会专利商标事务所有限公司 11038
专利代理师 张鑫

(51)Int.Cl.
G06F 40/10(2020.01)
H04L 67/01(2022.01)
G06F 16/332(2025.01)
G06F 16/35(2025.01)

(56)对比文件
US 2018113894 A1,2018.04.26
US 8856642 B1,2014.10.07

审查员 辛铁钢

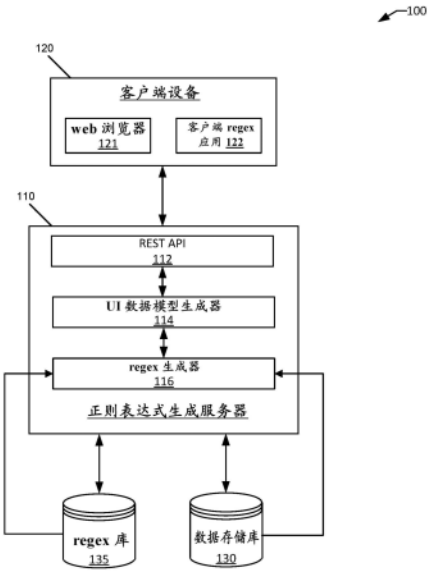
权利要求书4页 说明书32页 附图23页

(54)发明名称

基于肯定和否定模式匹配示例的正则表达式生成

(57)摘要

本文公开的是与正则表达式的自动生成相关的技术。在一些实施例中,正则表达式生成器可以接收包括一个或多个字符序列的输入数据。正则表达式生成器可以将字符序列转换成正则表达式代码和/或跨度数据结构的集合。正则表达式生成器可以标识由正则表达式代码和/或跨度的集合共享的最长通用子序列,并且可以基于最长通用子序列来生成正则表达式。



CN 112262390 B

1.一种生成正则表达式的方法,包括:

由包括一个或多个处理器的正则表达式生成器接收包括一个或多个肯定字符序列的第一输入数据,所述一个或多个肯定字符序列中的每一个对应于要与所述正则表达式生成器生成的正则表达式匹配的肯定示例;

由所述正则表达式生成器生成第一正则表达式,其中所述第一正则表达式与所述一个或多个肯定示例中的每一个匹配;

由所述正则表达式生成器接收包括一个或多个否定字符序列的第二输入数据,所述一个或多个否定字符序列中的每一个对应于要与所述正则表达式生成器生成的正则表达式不匹配的否定示例;

响应于接收到所述第二输入数据,确定所述一个或多个否定示例中的每一个是否与所述第一正则表达式匹配;以及

响应于确定至少一个否定示例与所述第一正则表达式匹配:

(a)确定所述第一正则表达式内的位置处的字符子序列;

(b)确定替换字符序列,其中所述正则表达式内的所述位置处的替换字符序列区分所述一个或多个肯定示例与所述一个或多个否定示例;以及

(c)通过用所述替换字符序列替换所述第一正则表达式内的所确定的字符子序列来更新所述第一正则表达式。

2.如权利要求1所述的方法,其中确定所述第一正则表达式内的所述位置处的字符子序列包括:

确定所述第一正则表达式内的所述位置;

从所述一个或多个肯定示例中的每一个以及从所述一个或多个否定示例中的每一个检索对应于所述第一正则表达式内的所述位置的文本片段;以及

将所述字符子序列确定为所述第一正则表达式内的所述位置处的能够区分所述一个或多个肯定示例和所述一个或多个否定示例的一个或多个字符。

3.如权利要求2所述的方法,其中确定所述第一正则表达式内的所述位置包括:

确定所述第一正则表达式的前缀部分处的能够区分所述一个或多个肯定示例和所述一个或多个否定示例的第一数量的字符;

确定所述第一正则表达式的后缀部分处的能够区分所述一个或多个肯定示例和所述一个或多个否定示例的第二数量的字符;以及

至少部分地基于所述第一数量的字符还是所述第二数量的字符较短来选择所述前缀部分或是所述后缀部分作为所述第一正则表达式内的所述位置。

4.如权利要求3所述的方法,其中确定所述第一正则表达式内的所述位置还包括:

执行公式以确定所述第一正则表达式内的所述位置,其中所述公式将所述第一正则表达式的所述前缀部分相对于所述后缀部分加重。

5.如权利要求2所述的方法,其中所述第一正则表达式内的所确定的位置是不对应于所述第一正则表达式的前缀部分或所述第一正则表达式的后缀部分的中间跨度位置。

6.如权利要求2所述的方法,其中确定所述替换字符序列包括确定多个替换字符序列,并且其中更新所述第一正则表达式包括用所述多个替换字符序列替换所述第一正则表达式内的所确定的字符子序列。

7.如权利要求1所述的方法,其中确定所述替换字符序列包括:

确定所述第一正则表达式内的所述位置处的能够区分所述一个或多个肯定示例和所述一个或多个否定示例的第一数量的字符以及相应的第一数量的替换字符序列,所述相应的第一数量的替换字符序列中的每个替换字符序列具有所述第一数量的字符;

确定所述第一正则表达式内的所述位置处的能够区分所述一个或多个肯定示例和所述一个或多个否定示例的第二数量的字符以及相应的第二数量的替换字符序列,所述相应的第二数量的替换字符序列中的每个替换字符序列具有所述第二数量的字符;以及

基于(a)所述第一数量的字符和所述第二数量的字符的大小以及(b)所述相应的第一数量的替换字符序列和所述相应的第二数量的替换字符序列的大小,选择所述第一数量的字符或所述第二数量的字符用于所述第一正则表达式内的替换字符序列。

8.一种用于生成正则表达式的系统,所述系统包括:

包括一个或多个处理器的处理单元;以及

存储指令的存储器,所述指令当由所述处理单元执行时使得所述系统:

接收包括一个或多个肯定字符序列的第一输入数据,所述一个或多个肯定字符序列中的每一个对应于要与所述正则表达式生成器生成的正则表达式匹配的肯定示例;

生成第一正则表达式,其中所述第一正则表达式与所述一个或多个肯定示例中的每一个匹配;

接收包括一个或多个否定字符序列的第二输入数据,所述一个或多个否定字符序列中的每一个对应于要与所述正则表达式生成器生成的正则表达式不匹配的否定示例;

响应于接收到所述第二输入数据,确定所述一个或多个否定示例中的每一个是否与所述第一正则表达式匹配;以及

响应于确定至少一个否定示例与所述第一正则表达式匹配:

(a)确定所述第一正则表达式内的位置处的字符子序列;

(b)确定替换字符序列,其中所述正则表达式内的所述位置处的替换字符序列区分所述一个或多个肯定示例与所述一个或多个否定示例;以及

(c)通过用所述替换字符序列替换所述第一正则表达式内的所确定的字符子序列来更新所述第一正则表达式。

9.如权利要求8所述的系统,其中确定所述第一正则表达式内的所述位置处的字符子序列包括:

确定所述第一正则表达式内的所述位置;

从所述一个或多个肯定示例中的每一个以及从所述一个或多个否定示例中的每一个检索对应于所述第一正则表达式内的所述位置的文本片段;以及

将所述字符子序列确定为所述第一正则表达式内的所述位置处的能够区分所述一个或多个肯定示例和所述一个或多个否定示例的一个或多个字符。

10.如权利要求9所述的系统,其中确定所述第一正则表达式内的所述位置包括:

确定所述第一正则表达式的前缀部分处的能够区分所述一个或多个肯定示例和所述一个或多个否定示例的第一数量的字符;

确定所述第一正则表达式的后缀部分处的能够区分所述一个或多个肯定示例和所述一个或多个否定示例的第二数量的字符;以及

至少部分地基于所述第一数量的字符还是所述第二数量的字符较短来选择所述前缀部分或是所述后缀部分作为所述第一正则表达式内的所述位置。

11.如权利要求10所述的系统,其中确定所述第一正则表达式内的所述位置还包括:

执行公式以确定所述第一正则表达式内的所述位置,其中所述公式将所述第一正则表达式的所述前缀部分相对于所述后缀部分加重。

12.如权利要求9所述的系统,其中所述第一正则表达式内的所确定的位置是不对应于所述第一正则表达式的前缀部分或所述第一正则表达式的后缀部分的中间跨度位置。

13.如权利要求9所述的系统,其中确定所述替换字符序列包括确定多个替换字符序列,并且其中更新所述第一正则表达式包括用所述多个替换字符序列替换所述第一正则表达式内的所确定的字符子序列。

14.如权利要求8所述的系统,其中确定所述替换字符序列包括:

确定所述第一正则表达式内的所述位置处的能够区分所述一个或多个肯定示例和所述一个或多个否定示例的第一数量的字符以及相应的第一数量的替换字符序列,所述相应的第一数量的替换字符序列中的每个替换字符序列具有所述第一数量的字符;

确定所述第一正则表达式内的所述位置处的能够区分所述一个或多个肯定示例和所述一个或多个否定示例的第二数量的字符以及相应的第二数量的替换字符序列,所述相应的第二数量的替换字符序列中的每个替换字符序列具有所述第二数量的字符;以及

基于(a)所述第一数量的字符和所述第二数量的字符的大小以及(b)所述相应的第一数量的替换字符序列和所述相应的第二数量的替换字符序列的大小,选择所述第一数量的字符或所述第二数量的字符用于所述第一正则表达式内的替换字符序列。

15.一种用于生成正则表达式的非暂态计算机可读介质,所述计算机可读介质包括计算机可执行指令,所述计算机可执行指令在计算机系统上执行时使得所述计算机系统:

接收包括一个或多个肯定字符序列的第一输入数据,所述一个或多个肯定字符序列中的每一个对应于要与所述正则表达式生成器生成的正则表达式匹配的肯定示例;

生成第一正则表达式,其中所述第一正则表达式与所述一个或多个肯定示例中的每一个匹配;

接收包括一个或多个否定字符序列的第二输入数据,所述一个或多个否定字符序列中的每一个对应于要与所述正则表达式生成器生成的正则表达式不匹配的否定示例;

响应于接收到所述第二输入数据,确定所述一个或多个否定示例中的每一个是否与所述第一正则表达式匹配;以及

响应于确定至少一个否定示例与所述第一正则表达式匹配:

(a)确定所述第一正则表达式内的位置处的字符子序列;

(b)确定替换字符序列,其中所述正则表达式内的所述位置处的替换字符序列区分所述一个或多个肯定示例与所述一个或多个否定示例;以及

(c)通过用所述替换字符序列替换所述第一正则表达式内的所确定的字符子序列来更新所述第一正则表达式。

16.如权利要求15所述的计算机可读介质,其中确定所述第一正则表达式内的所述位置处的字符子序列包括:

确定所述第一正则表达式内的所述位置;

从所述一个或多个肯定示例中的每一个以及从所述一个或多个否定示例中的每一个检索对应于所述第一正则表达式内的所述位置的文本片段;以及

将所述字符子序列确定为所述第一正则表达式内的所述位置处的能够区分所述一个或多个肯定示例和所述一个或多个否定示例的一个或多个字符。

17.如权利要求16所述的计算机可读介质,其中确定所述第一正则表达式内的所述位置包括:

确定所述第一正则表达式的前缀部分处的能够区分所述一个或多个肯定示例和所述一个或多个否定示例的第一数量的字符;

确定所述第一正则表达式的后缀部分处的能够区分所述一个或多个肯定示例和所述一个或多个否定示例的第二数量的字符;以及

至少部分地基于所述第一数量的字符还是所述第二数量的字符较短来选择所述前缀部分或是所述后缀部分作为所述第一正则表达式内的所述位置。

18.如权利要求17所述的计算机可读介质,其中确定所述第一正则表达式内的所述位置还包括:

执行公式以确定所述第一正则表达式内的所述位置,其中所述公式将所述第一正则表达式的所述前缀部分相对于所述后缀部分加重。

19.如权利要求16所述的计算机可读介质,其中所述第一正则表达式内的所确定的位置是不对应于所述第一正则表达式的前缀部分或所述第一正则表达式的后缀部分的中间跨度位置。

20.如权利要求16所述的计算机可读介质,其中确定所述替换字符序列包括确定多个替换字符序列,并且其中更新所述第一正则表达式包括用所述多个替换字符序列替换所述第一正则表达式内的所确定的字符子序列。

21.一种计算机程序产品,包括指令,所述指令在由计算机执行时,使得所述计算机执行如权利要求1-7中任一项所述的方法。

基于肯定和否定模式匹配示例的正则表达式生成

[0001] 相关申请交叉引用

[0002] 本申请根据35U.S.C.§119(e)要求于2018年6月13日提交的、题为“AUTOMATED GENERATION OF REGULAR EXPRESSIONS”的美国临时专利申请号62/684,498的优先权,并且本申请还根据35U.S.C.§119(e)要求于2018年10月22日提交的、题为“AUTOMATED GENERATION OF REGULAR EXPRESSIONS”的美国临时专利申请号62/749,001的优先权。美国临时专利申请号62/684,498和62/749,001的全部内容通过引用合并于此以供用于所有目的。

背景技术

[0003] 大数据分析系统可用于预测性分析、用户行为分析和其他高级数据分析。然而,在有效执行任何数据分析以提供有用结果之前,可能需要将初始数据集格式化为干净和精选的数据集。这种数据加载通常给基于云的数据存储库和其他大数据系统带来挑战,在这些系统中,来自各种不同数据源和/或数据流的数据可以被编译到单个数据存储库中。这种数据可以包括以多种不同格式的结构化数据、根据不同数据模型的半结构化数据、甚至非结构化数据。这种数据的存储库通常包括各种不同格式和结构的数据表示,并且还可以包括重复数据和错误数据。在分析这些数据存储库以进行报告、预测性建模和其他分析任务时,初始数据集的较差信噪比可能会导致不准确或无用的结果。

[0004] 对数据格式化和预处理问题的许多当前解决方案包括手动和adhoc处理以清理和整理数据,以便在执行数据分析之前将数据处理成通用格式。虽然这些手动过程对于某些较小的数据集可能是有效的,但是当试图对大规模数据集进行预处理和格式化时,这样的过程可能效率低下且不切实际。

发明内容

[0005] 本文描述的各方面提供了用于生成正则表达式的各种技术。如本文所使用的,“正则表达式”可指定义模式的字符序列,其可用于在较长的输入文本串内搜索匹配。在一些实施例中,可以使用符号通配符匹配语言来组成正则表达式,并且由正则表达式定义的模式可以用于匹配字符串和/或从作为输入提供的字符串中提取信息。在本文描述的各种实施例中,实现为数据处理系统的正则表达式生成器可用于接收和显示输入文本数据,经由客户端用户界面接收对输入文本的特定字符子集的选择,然后基于所选择的字符子集生成一个或多个正则表达式。在生成一个或多个正则表达式之后,可以使用正则表达式引擎来将正则表达式的模式与一个或多个数据集进行匹配。在各种实施例中,与正则表达式匹配的数据可以被提取、重新格式化或修改等等。在一些情况下,可以基于与正则表达式匹配的数据来创建附加的列、表或其他数据集。

[0006] 根据本文描述的某些方面,经由数据处理系统实现的正则表达式生成器可以基于由一个或多个正则表达式代码的不同集合共享的所确定的最长通用子序列(LCS)来生成正则表达式。正则表达式代码(也可以称为类别代码)可以包括,例如,L代表英文字母表中的

字母,N代表数字,Z代表空格,P代表标点符号,以及S代表其他符号。一个或多个正则表达式代码的每个集合可以从作为输入数据通过用户界面接收的一个或多个字符的不同序列转换。从LCS中排除的正则表达式代码可以被表示为可选的和/或备选的。在一些实施例中,正则表达式代码可以与该正则表达式代码的最小出现次数相关联。附加地或备选地,正则表达式代码可以与正则表达式代码的最大出现次数相关联。例如,类别代码集合可以包括L<0,1>,以指示LCS的特定部分包括字母最多一次(如果有的话)。如下面更详细讨论的,将输入数据概括为中间正则表达式代码(IREC)可以提供各种技术优势,包括使用非常少的输入数据,使得能够近乎即时地生成不被尚未看到的数据中的假阳性匹配或假阴性匹配影响的正则表达式。

[0007] 根据本文描述的附加方面,可以基于包括三个或更多个字符序列的输入数据来生成正则表达式。当三个或更多字符序列被标识为输入数据时,标识字符序列的LCS的正则表达式生成器可能导致运行时间的指数增加。为了以高性能方式标识所有字符序列的LCS,正则表达式生成器可以对两个字符序列的每个不同组合执行LCS算法。可以基于LCS算法的结果生成全连接的图,其中每个图节点代表不同的字符序列,并且每个图边的长度对应于定义图边的节点的LCS。然后,可以通过对全连接图的最小生成树执行深度优先遍历来确定选择字符序列的顺序。

[0008] 本文描述的其他方面涉及基于包括肯定字符序列示例和否定字符序列示例两者的输入来生成正则表达式。肯定示例可以指与要生成的正则表达式匹配的字符序列,而否定示例可以指与要生成的正则表达式不匹配的字符序列。在一些实施例中,当接收到肯定示例和否定示例两者时,正则表达式生成器可以识标识将肯定示例和否定示例区分开的一个或多个字符的鉴别符或最短子序列。所选择的鉴别符可以是最短序列(例如,以类别代码表示),并且可以是肯定的或否定的,使得肯定示例将匹配而否定示例将不匹配。然后,可以将鉴别符硬编码到由正则表达式生成器生成的正则表达式中。在一些情况下,最短子序列可以包括在否定示例的前缀或后缀部分中。

[0009] 本文描述的附加方面涉及可通过其提供输入数据以生成正则表达式的一个或多个用户界面。在一些实施例中,可以在通信地耦合到正则表达式生成器服务器的客户端设备处显示用户界面。用户界面可以由服务器、由客户端设备、或由在服务器和客户端执行的软件组件的组合以编程方式生成。通过用户界面接收的输入数据可以对应于用户对一个或多个字符序列的选择,这些字符序列可以代表肯定或否定示例。在一些情况下,用户界面可以支持包括在第二字符序列内对第一字符序列的选择的输入数据。例如,用户可以突出显示较大的先前突出显示的字符序列中的一个或多个字符,并且第二用户选择可以针对较大的第一用户选择提供上下文。这使得输入数据能够更有针对性地被提供给正则表达式生成器,并且为正则表达式生成器提供“上下文”,使得它可以生成避免假阳性的正则表达式。响应于用户经由用户界面选择字符序列,正则表达式生成器可以生成并显示正则表达式。例如,当用户突出显示第一字符序列时,正则表达式生成器可以生成并显示匹配第一字符序列以及其他相似字符序列(例如,与用户针对匹配序列的意图一致)的正则表达式。当用户突出显示第二字符序列时,正则表达式生成器可以生成包含第一和第二字符序列的更新的正则表达式。然后,当用户突出显示第三字符序列(例如,在第一或第二序列内)时,正则表达式生成器可以再次更新正则表达式,以此类推。

[0010] 根据本文描述的附加方面,正则表达式可以基于来自一个或多个输入序列示例的最长通用子序列来生成,也可以处理仅在一些示例中出现的字符。为了处理仅在一些输入示例中出现的字符,可以定义跨度,其中追踪正则表达式代码的最小和最大出现次数。在跨度可能不存在于所有给定输入示例的情况下,可以将最小出现次数设置为零。然后可以将这些最小和最大数字映射到正则表达式重度语法。最长通用子序列(LCS)算法可以在从输入示例得到的字符的跨度上运行,包括并非在每个输入示例中都出现的"可选的"跨度(例如,最小长度为零)。如下文所讨论的,可以在LCS算法的执行期间合并连续跨度。在这种情况下,当携带的额外可选跨度不再连续出现时,LCS算法也可以在这些可选跨度上递归运行。

[0011] 本文描述的其他方面涉及组合搜索,其中由正则表达式生成器执行的LCS算法可以被多次运行以生成"正确的"正则表达式(例如,与所有给定的肯定示例正确匹配并且适当地排除所有给定的否定示例的正则表达式),和/或生成多个正确的正则表达式,从中可以选择最期望的或最佳的正则表达式。在一些实施例中,通常可以对输入示例从右向左执行LCS算法以生成正则表达式。然而,出于比较的目的并为了找到备选的正则表达式,LCS算法可以在输入示例上向后(例如,在从左到右的方向上)分别执行。例如,作为用户输入接收的示例字符序列可以在它们运行LCS算法之前被颠倒,然后来自LCS算法的结果可以被颠倒回来(包括原始文本片段)。此外,在一些实施例中,LCS算法可以由正则表达式生成器运行多次,以通常的字符序列顺序和相反的顺序,在行首锚定,在行尾锚定,在行首或行尾不锚定。因此,在一些情况下,LCS算法可以至少执行这样六次,并且可以从这些执行中选择最短的成功正则表达式。

附图说明

[0012] 图1是示出用于生成正则表达式的示例性分布式系统的组件的框图,其中可以实现各种实施例。

[0013] 图2是根据本文描述的一个或多个实施例,示出基于经由用户界面接收的输入来生成正则表达式的过程的流程图。

[0014] 图3是根据本文描述的一个或多个实施例,示出在正则表达式代码集合上使用最长通用子序列(LCS)算法来生成正则表达式的过程的流程图。

[0015] 图4是根据本文描述的一个或多个实施例,基于两个字符序列示例、在正则表达式代码集合上使用最长通用子序列(LCS)算法来生成正则表达式的示例图。

[0016] 图5是根据本文描述的一个或多个实施例,示出在较大的正则表达式代码集合上使用最长通用子序列(LCS)算法来生成正则表达式的过程的流程图。

[0017] 图6是根据本文描述的一个或多个实施例,基于五个字符序列示例、在正则表达式代码集合上使用最长通用子序列(LCS)算法来生成正则表达式的示例图。

[0018] 图7是根据本文描述的一个或多个实施例,示出用于在较大的正则表达式代码集合上确定针对最长通用子序列(LCS)算法的执行顺序的过程的流程图。

[0019] 图8A和8B根据本文描述的一个或多个实施例,示出了全连接图和全连接图的最小生成树表示,用于在较大的正则表达式代码集合上确定针对最长通用子序列(LCS)算法的执行顺序。

[0020] 图9是根据本文描述的一个或多个实施例,示出基于肯定和否定字符序列示例来生成正则表达式的过程的流程图。

[0021] 图10A和10B是根据本文描述的一个或多个实施例,示出基于肯定和否定字符序列示例生成正则表达式的示例用户界面屏幕。

[0022] 图11是根据本文描述的一个或多个实施例,示出基于在用户界面内接收的用户数据选择来生成正则表达式的过程的流程图。

[0023] 图12是根据本文描述的一个或多个实施例,示出用于基于捕获组,经由在用户界面内接收的用户数据选择生成正则表达式和提取数据的流程图。

[0024] 图13是根据本文描述的一个或多个实施例,示出表格数据显示的示例用户界面屏幕。

[0025] 图14和15是根据本文描述的一个或多个实施例,示出基于来自表格显示的数据选择来生成正则表达式和捕获组的示例用户界面屏幕。

[0026] 图16A和16B是根据本文描述的一个或多个实施例,示出基于来自表格显示的肯定和否定示例选择来生成正则表达式的示例用户界面屏幕。

[0027] 图17是根据本文描述的一个或多个实施例,示出基于来自表格显示的数据选择来生成正则表达式和捕获组的另一示例用户界面屏幕。

[0028] 图18是根据本文描述的一个或多个实施例,示出使用最长通用子序列(LCS)算法生成包括可选跨度的正则表达式的过程的流程图。

[0029] 图19是根据本文描述的一个或多个实施例,使用最长通用子序列(LCS)算法生成包括可选跨度的正则表达式的示例图。

[0030] 图20是根据本文描述的一个或多个实施例,示出基于最长通用子序列(LCS)算法的组合执行来生成正则表达式的过程的流程图。

[0031] 图21是示出可以实现本发明的各种实施例的示例性分布式系统的组件的框图。

[0032] 图22是示出可以将由本发明的实施例提供的服务提供为云服务的系统环境的组件的框图。

[0033] 图23是示出可以实现本发明的实施例的示例性计算机系统的框图。

具体实施方式

[0034] 在下面的描述中,出于解释的目的,呈现了许多具体细节,以便提供对本发明的各种实施例的透彻理解。然而,对于本领域技术人员来说显而易见的是,本发明的实施例可以在没有这些具体细节的情况下实施。在其他情况下,公知的结构和设备以框图形式示出。

[0035] 下面的描述仅提供示例性实施例,并不意图限制本公开的范围、适用性或配置。相反,下面对示例性实施例的描述将向本领域技术人员提供用于实现示例性实施例的实现描述。应当理解,在不背离所附权利要求中陈述的本发明的精神和范围的情况下,可以在元件的功能和布置上进行各种改变。

[0036] 在下面的描述中给出具体细节以提供对实施例的透彻理解。然而,本领域普通技术人员将理解,可以在没有这些具体细节的情况下实施实施例。例如,电路、系统、网络、过程和其他组件可被示为框图形式的组件,以便不在不必要的细节中混淆实施例。在其他情况下,为了避免混淆实施例,可以在没有不必要细节的情况下示出公知的电路、过程、算法、

结构和技术。

[0037] 还要注意的是,各个实施例可以被描述为过程,该过程被描绘为流程图表、流程图、数据流程图、结构图或框图。虽然流程图可以将操作描述为顺序过程,但许多操作可以并行或并发执行。此外,可以重新安排操作的顺序。过程在其操作完成时终止,但可能有图中未包括的其他步骤。过程可以对应于方法、函数、进程、子例程、子程序等。当过程对应于函数时,其终止可以对应于函数返回到调用函数或主函数。

[0038] 术语“计算机可读介质”包括但不限于诸如便携式或固定存储设备的非暂态介质、光学存储设备、以及能够存储、包含或携带指令和/或数据的各种其他介质。代码段或计算机可执行指令可以代表进程、函数、子程序、程序、例程、子例程、模块、软件包、类或指令、数据结构或程序语句的任意组合。代码段可以通过传递和/或接收信息、数据、自变量、参数或存储器内容而耦合到另一代码段或硬件电路。信息、自变量、参数、数据等可以经由包括存储器共享、消息传递、令牌传递、网络传输等任何合适的手段来传递、转发或传输。

[0039] 此外,实施例可以通过硬件、软件、固件、中间件、微码、硬件描述语言或其任意组合来实现。当以软件、固件、中间件或微码实现时,用于执行必要任务的程序代码或代码段可以被存储在机器可读介质中。处理器可以执行必要任务。

[0040] 本文描述了用于生成对应于在一个或多个输入数据示例中标识的模式的正则表达式的各种技术(例如,方法、系统、存储可由一个或多个处理器执行的多个指令的非暂态计算机可读存储器等)。在某些实施例中,响应于接收到对输入数据的选择,自动标识输入数据中的一个或多个模式,并且可以自动且高效地生成正则表达式(或简称“regex”)以表示所标识的模式。这种模式可以基于字符序列(例如,字母、数字、空格、标点符号、符号等的序列)。本文描述了各种实施例,包括方法、系统、存储可由一个或多个处理器执行的程序、代码或指令的非暂态计算机可读存储介质等。

[0041] 在一些实施例中,可以使用符号通配符匹配语言来组成正则表达式,以便匹配字符串和/或从作为输入提供的字符串中提取信息。例如,第一示例正则表达式[A-Za-z]{3}\d?\d,\d\d\d\d可以匹配某些日期(例如,2018年4月3日),并且第二示例正则表达式[A-Za-z]{3}\d?\d,(\d\d\d\d)可以用于从匹配日期中提取年份。正则表达式生成器系统接收的输入数据可以包括,例如,一个或多个“肯定”数据示例,和/或一个或多个“否定”数据示例。如本文所使用的,肯定示例可以指作为输入接收的,要与基于该输入生成的正则表达式匹配的字符序列。相反,否定示例可以指要与基于该输入生成的正则表达式不匹配的输入字符序列。

[0042] 在本文描述的各种实施例和示例内可以实现许多技术优势。例如,本公开中描述的某些技术可以提高正则表达式生成过程的速度和效率(例如,regex解决方案可以在不到一秒内生成,并且用户界面可以适合于交互式实时使用)。本文描述的各种技术也可以是确定性的,可以不需要训练数据,可以在不需要任何初始正则表达式输入的情况下产生解决方案,并且可以完全自动化(例如,在不需要任何人工干预的情况下生成正则表达式)。此外,本文描述的各种技术不需要限制可以被有效处理的数据输入的类型,并且这样的技术可以提高所得到的正则表达式的人类可读性。

[0043] 本文描述的某些实施例包括最长通用子序列(LCS)算法的一个或多个执行。LCS算法可以在某些上下文中用作差异引擎(例如,Unix“diff”实用程序背后的引擎),其被配置

为确定并显示两个文本文件之间的差异。在一些实施例中,可以将输入数据(例如,字符串和其他字符序列)转换成抽象令牌,然后可以将抽象令牌作为输入提供给LCS算法。这样的抽象令牌可以是例如基于表示正则表达式字符类的正则表达式代码(例如,Loogle代码或其他字符类代码)的令牌。这种代码的各种不同示例是可能的,并且本文可以被称为"正则表达式代码"或"中间正则表达式代码"(IREC)。例如,可以将输入字符序列"May 3"转换成IREC码"LLLZN",之后可以向LCS算法提供该令牌化的字符串以及其他令牌化的字符串。在一些实施例中,输入字符序列不通用拥有的IREC(例如,正则表达式代码)可以作为可选(例如,可选的跨度)出现在最终生成的正则表达式中。在某些实施例中,正则表达式代码可以是基于https://www.regular-expressions.info/unicode.html#category所示的UNICODE类别代码的类别代码。例如,代码L可以代表字母,代码N可以代表数字,代码Z可以代表空格,代码S可以代表符号,代码P可以代表标点符号,等等。例如,代码L可以对应于Unicode\p{L},而代码N可以对应于Unicode\p{N}。这允许从LCS输出到正则表达式的一对一映射(例如,\pN\pN\pZ\pL\pL可以匹配"10am"),这可能有利于人类可读性。此外,这些不同的类别可能是不相交的,也可能是相互排斥的。也就是说,在该示例中,类别L、N、Z、P和S可以是不相交的,使得类别的成员之间可以不存在重叠。

[0044] 在各种实施例中可以实现附加的技术优势,包括基于正则表达式代码(例如,类别代码)、跨度等的使用来更有效地生成正则表达式。通过使用这样的代码,当LCS算法成功地将输入字符串中的所有或基本上所有字符标识为不同时,不需要浪费计算资源。本文的各种实施例提供的其他技术优势包括改进的所生成的正则表达式的可读性,以及支持作为输入数据的肯定示例和否定示例,以及提供各种有利的用户界面特征(例如,允许用户突出显示较大字符序列或数据单元内的文本片段以供提取)。

[0045] I.总体概述

[0046] 本文公开的各种实施例涉及正则表达式的生成。在一些实施例中,被配置为正则表达式生成器的数据处理系统可以通过标识由不同的正则表达式代码(例如,类别代码)的集合共享的最长通用子序列(LCS)来生成正则表达式。每个正则表达式代码集合可以从作为输入数据通过用户界面接收的字符序列转换。在本文描述的技术优势中,将输入数据抽象为中间代码(例如,正则表达式代码、跨度等)可以实现使用非常少的输入数据高效地生成正则表达式。

[0047] 图1是示出用于生成正则表达式的示例性分布式系统的组件的框图,其中可以实现各种实施例。如该示例中所示,客户端设备120可以与正则表达式生成器服务器110(或正则表达式生成器)通信,并与用户界面交互以检索和显示表格数据,并基于经由用户界面对输入数据(例如,示例)的选择来生成正则表达式。在一些实施例中,客户端设备120可以经由客户端web浏览器121和/或客户端侧正则表达式应用122(例如,接收/消费由服务器110生成的正则表达式的客户端应用)与正则表达式生成器110通信。在正则表达式生成器110内,来自客户端设备120的请求可以在网络接口处通过各种通信网络被接收,并由诸如REST API 112的应用编程接口(API)来处理。具有正则表达式生成器110的用户界面数据模型生成器114组件可以提供服务器侧编程组件和逻辑,以生成和呈现本文描述的各种用户界面特征。这样的特征可以包括以下功能性:允许用户从数据存储库130检索和显示表格数据,选择输入数据示例以启动正则表达式的生成,以及基于所生成的正则表达式修改和/或提

取数据。在该示例中,正则表达式生成器组件116可以被实现为生成正则表达式,包括将输入字符序列转换成正则表达式代码和/或跨度,对输入数据执行算法(例如,LCS算法)以及生成/简化正则表达式。由正则表达式生成器116生成的正则表达式可以由REST服务112传输到客户端设备120,在客户端设备120中,客户端浏览器121(或相应的客户端应用组件122)上的Javascript代码然后可以针对在浏览器中呈现的电子表格列中的每个单元格应用正则表达式。在其他情况下,可以在服务器侧实现单独的正则表达式引擎组件,以将生成的正则表达式与用户界面上显示的表格数据和/或存储在数据存储库130中的其他数据进行比较,以便标识服务器侧上的匹配数据/不匹配数据。在各种实施例中,匹配/非匹配数据可以在用户界面内被自动选择(例如,突出显示),并且可以被选择用于提取、修改、删除等。基于正则表达式的生成,经由用户界面提取或修改的任何数据可以被存储在一个或多个数据存储库130中。此外,在一些实施例中,所生成的正则表达式(和/或LCS算法的相应输入)可以存储在正则表达式库135中以供将来检索和使用。在一些实施例中,所生成的正则表达式实际上不需要存储在“库”中,而是可以合并到“转换脚本”中。例如,如在美国专利号10,210,246号中更详细地描述的(其通过引用合并于此以用于所有目的),这样的转换脚本可以包括可由一个或多个处理单元执行以转换接收数据的程序、代码或指令。变换脚本动作的其他可能示例可以包括“重命名列”、“大写列数据”或“从名字推断性别并创建具有性别的新列”等。

[0048] 图2是根据本文描述的一个或多个实施例,示出基于经由用户界面接收的输入来生成正则表达式的过程的流程图。在步骤201中,正则表达式生成器110可以从客户端设备120接收访问正则表达式生成器用户界面并经由该用户界面查看特定数据的请求。可以经由REST API 112和/或web服务器、认证服务器等接收步骤201中的请求,并且可以解析和认证用户的请求。例如,企业或组织内的用户可以访问正则表达式生成器110以分析和/或修改交易数据、客户数据、绩效数据、预测数据和/或可以存储在组织的数据存储库130中的任何其他类别的数据。在步骤202中,正则表达式生成器110可以经由支持基于所选输入数据生成正则表达式的用户界面来检索和显示所请求的数据。下面详细描述这样的用户界面的各种实施例和示例。

[0049] 在步骤203中,用户可以从正则表达式生成器110提供的,用户界面中显示的数据中选择一个或多个输入字符序列。在一些实施例中,数据可以在用户界面内以表格形式显示,包括具有特定数据类型和/或数据类别的标签化列。在这种情况下,在步骤203中对输入数据的选择可以对应于用户选择数据单元格,或者选择(例如,突出显示)数据单元格内的单个文本片段。然而,在其他实施例中,正则表达式生成器110可以支持经由用户界面检索和显示半结构化和非结构化数据,并且用户可以通过从半结构化或非结构化数据中选择字符序列来选择用于正则表达式生成的输入数据。如下面在示例中描述的,用户从显示的表格数据中选择输入字符序列仅是一个示例用例。在其他示例中,用户(例如,软件开发人员或高级用户可能试图针对Linux命令行工具grep、sed或awk等编写正则表达式)可以从零开始输入示例,而不是从电子表格中摘取它们。

[0050] 在步骤204中,正则表达式生成器110可以基于用户在步骤203中选择的输入数据来生成一个或多个正则表达式。在步骤205中,正则表达式生成器110可以更新用户界面,例如,以显示所生成的正则表达式和/或突出显示所显示的数据内的匹配/不匹配数据。在步

骤206(在一些实施例中可以是可选的)中,用户界面可以支持允许用户基于所生成的正则表达式修改底层数据的功能。例如,用户界面可以支持如下特征:允许用户基于这些字段是否与正则表达式匹配来从表格数据中过滤、修改、删除或提取特定数据字段。过滤或修改数据可以包括修改存储在存储库130中的底层数据,并且在某些情况下,所提取的数据可以作为新列和/或新表格被存储在存储库130中。

[0051] 尽管这些步骤示出了与正则表达式生成器110的用户界面的示例性用户交互的一般和高级概述,但是在其他实施例中可以支持各种附加特征和功能。例如,在一些实施例中,正则表达式代码(或类别代码)可以与该代码的最小出现次数相关联。附加地或备选地,正则表达式代码可以与代码的最大出现次数相关联。作为示例,正则表达式代码的集合可以包括代码L<0,1>,以指示LCS的特定部分包括字母至少零次且至多一次。

[0052] 此外,在一些实施例中,输入数据可以包括三个或更多个字符序列。在这样的实施例中,可以使用技术来确定对三个或更多个字符序列执行LCS算法的顺序,从而可以以高性能方式生成所得到的正则表达式,以避免由三个或更多个输入字符序列导致的运行时间的指数增加。相反,正则表达式生成器110可以一次对两个字符序列执行LCS算法,并且可以基于图来确定用于选择这对字符序列的顺序。例如,全连接图可以指示应针对序列1和序列3执行LCS算法(例如,LCS1)的第一执行,然后应针对LCS1和序列2执行LCS算法(例如,LCS2)的第二执行,以此类推。该图可以是全连接图,具有表示字符序列的节点,以及连接节点的边以表示由所连接的节点共享的LCS的长度。图中的每个节点可以连接到图中的每个其他节点,并且可以通过执行针对图的最小生成树的深度优先遍历来确定用于选择字符序列的顺序。

[0053] 在其他实施例中,可以经由用户界面以多种不同方式提供输入数据。例如,输入数据可以指示在字符集的第二用户选择内对一个或多个字符的第一用户选择。例如,用户可以突出显示先前突出显示的字符集合中的字符。因此,第二用户选择可以提供针对第一用户选择的上下文,这可以使输入数据能够更有针对性地被提供给正则表达式生成器110。在一些实施例中,正则表达式生成器110可以响应于每个用户选择近乎实时地生成并显示正则表达式。例如,当用户突出显示第一字符范围时,正则表达式生成器110可以显示表示第一字符范围的正则表达式。然后,当用户突出显示第一字符范围内的第二字符范围时,正则表达式生成器110可以更新所显示的正则表达式。

[0054] 另外,在一些实施例中,正则表达式生成器110可以基于包括肯定和否定示例的输入来生成正则表达式。如上所述,肯定示例可以指要被正则表达式包含的字符序列,而否定示例可以指将不被正则表达式包含的字符序列。在这种情况下,正则表达式生成器110可以在特定位置标识将肯定示例和否定示例区分开的一个或多个字符的最短子序列。然后,最短子序列可以硬编码在正则表达式生成器110生成的正则表达式内。在各种示例中,最短子序列可以包括在否定示例内的前缀/后缀部分或中间跨度内。

[0055] 下面描述根据某些实施例的用于自动生成正则表达式的其他示例。这些示例可以对应于图2中的通用技术的各种具体可能的实现,并且可以用由相应系统、硬件或其组合的一个或多个处理单元(例如,处理器、核)执行的软件(例如,代码、指令、程序等)来实现。软件可以存储在非易失性存储介质上(例如,存储设备上)。下面描述的其他示例意在是说明性的和非限制性的。虽然这些示例描述了以特定序列或顺序发生的各种处理步骤,但这并

不意图是限制性。在某些备选实施例中,这些步骤可以按某种不同的顺序执行,或者某些步骤也可以并行执行。

[0056] 在一些示例中,经由用户界面接收的用户输入(例如,步骤203)可以包括正则表达式输出要匹配的一个或多个“肯定示例”,以及正则表达式输出不匹配的零个或多个“否定示例”。可选地,可以突出显示一个或多个肯定示例以选择特定范围(或子序列)的字符。在一些情况下,在步骤204中,可以将经由用户界面接收的肯定示例转换为正则表达式代码(例如,诸如Unicode类别代码的字符类别代码)的跨度。对于每个肯定示例,可以生成一系列跨度。在一些实施例中可以创建图,其中每个顶点对应于跨度的序列之一,并且边权重等于在对应于边的端点的这两个跨度序列上执行的LCS算法的输出的长度。可以针对该图确定最小生成树。例如,在一些实施例中可以使用Prim算法来获得最小生成树。可以在最小生成树上执行深度优先遍历以确定遍历顺序,之后可以在遍历的前两个元素上执行LCS算法。然后,通过对前一次LCS迭代的输出和下一次当前遍历元素再次执行LCS算法,可以一个接一个地将遍历的每个附加元素按顺序合并到当前LCS输出中。然后,可以将LCS算法的最终输出(其可以是跨度序列)转换为正则表达式。在一些实施例中,转换可以是一对一转换,而本文描述的某些可选修饰可能不对应于一对一转换。最后,在步骤203中,可以针对经由用户界面接收的所有肯定和否定示例来测试所得到的正则表达式。如果任何测试失败,则可以使用所有失败的肯定示例和任何否定示例重复上述过程。

[0057] II.在正则表达式代码上使用最长通用子序列算法的正则表达式生成

[0058] 如上所述,本文描述的某些方面涉及基于计算由对应于输入数据的不同正则表达式代码集合共享的最长通用子序列(LCS)来生成正则表达式。

[0059] 图3是根据本文描述的一个或多个实施例,示出在正则表达式代码集上使用LCS算法生成正则表达式的过程的流程图。在步骤301中,正则表达式生成器110可以接收一个或多个字符序列作为输入数据。如上所述,在一些示例中,输入数据可以对应于从用户界面中显示的表格数据中选择的肯定示例数据,但应该理解,在一些实施例中,用户界面是可选的,并且在各种示例中,输入数据可以对应于从任何其他通信信道(例如,非用户界面)接收的任何字符序列。

[0060] 在步骤302中,可以将在步骤301中接收到的每个字符序列转换为相应的正则表达式代码。在各种实施例中,正则表达式代码可以是Loogle码、Unicode类别代码或表示正则表达式字符类的任何其他字符类代码。例如,输入字符序列“May 3”可以被转换成Loogle码“LLLZN”。在一些实施例中,正则表达式代码可以是基于https://www.regular-expressions.info/unicode.html#category所示的UNICODE类别代码的类别代码。例如,代码L可以代表字母,代码N可以代表数字,代码Z可以代表空格,代码S可以代表符号,代码P可以代表标点符号,等等。例如,代码L可以对应于Unicode\p{L},而代码N可以对应于Unicode\p{N}。

[0061] 在步骤303中,可以根据在步骤302中生成的正则表达式代码集合确定最长通用子序列。在一些实施例中,可以使用两个正则表达式代码集合作为输入来执行LCS算法。可以在不同的实施例中使用LCS算法的执行的各种不同特征(例如,处理方向、锚定、推送空间、合并低基数跨度、对齐通用令牌等)。在步骤304中,可以基于LCS算法的输出来生成正则表达式。在一些情况下,步骤304可以包括捕获正则表达式代码中的LCS算法的输出,并将正则

表达式代码转换为正则表达式。在步骤305中,可以例如通过经由用户界面向用户显示正则表达式来简化并输出正则表达式。

[0062] 图4是用于基于两个字符序列示例,在正则表达式代码集合上使用最长通用子序列(LCS)算法来生成正则表达式的示例图。因此,图4示出了应用上面在图3中讨论的处理的示例。如图4所示,该示例中的正则表达式是基于两个输入串:“iPhone 5”和“iPhone X”生成的。该示例中的每个序列可以被转换成相应的正则表达式代码集合。因此,iPhone 5可以被转换成“LLLLLLZN”,iPhone X可以被转换成“LLLLLLZL”。如图4所示,然后将这些类别代码作为输入提供给LCS算法,该算法确定两个IREC(或类别代码)都包括六个L和一个Z。从LCS中排除的类别代码可以表示为可选的和/或备选的。因此,包含两个字符序列的正则表达式可以表示为:\pL{6}\pZ\pN?\pL?。在本示例中,正则表达式包括Unicode类别代码(例如,\pL代表字母,\pZ代表空格,\pN代表数字)。包含数字6的花括号指示字母的六个实例,问号指示末尾的数字/字母是可选的。最后,正则表达式生成器可以执行简化过程,简化过程期间通过将通用文本片段“iphone”重新插入回最终正则表达式中,替换正则表达式的更广泛的““\pL{6}\”部分,来简化正则表达式。

[0063] 如本示例所示,由正则表达式生成器110接收的输入串可以被转换成表示正则表达式广义类别(也可以被称为“类别代码”)的“正则表达式代码”,并且LCS算法可以在这些正则表达式代码上运行。在一些实施例中,Unicode类别代码可以用于正则表达式代码。例如,输入文本串可以被转换成表示regex Unicode广义类别的代码(例如,\pL代表字母,\pP代表标点符号等)。图3和图4所示的这种方法可以称为间接方法。然而,在其他实施例中,可以使用直接方法,其中LCS算法直接在作为输入接收的字符序列上运行。

[0064] 在一些实施例中,间接方法可以提供附加的技术优势,因为它不需要大量的训练数据,并且可以用相对较少的输入示例生成有效的正则表达式。这是因为间接方法采用启发式来减少正则表达式生成中的不确定性,并消除潜在的假阳性和假阴性。例如,在基于输入串“May 3”和“Apr 4”生成正则表达式时,直接方法可能每个月至少需要一个示例来生成与日期模式匹配的有效正则表达式。仅依靠这两个示例,直接方法可能会生成“[AM][ap][yr][13]1?”的regex。相反,基于Unicode广义类别,间接方法可以生成更有效的正则表达式“\pL{3}\d{1,2}”。此外,如上所述,本文描述的技术优势之一包括使用非常少的输入数据(甚至潜在地从单个示例)高效地生成正则表达式。例如,关于从单个示例“am”生成正则表达式,启发式可以确定针对正则表达式生成“am”还是“\pL\pL”。两者都可论证是正确的,但因此编程的启发式可以实现用户偏好和/或标准,以确定如何生成最佳正则表达式(例如,它是否也应该与“pm”匹配)。

[0065] 另外,间接方法还可以将所生成的正则表达式“\pL{3}\d{1,2}”简化为“[A-Za-z]{3}\d{1,2}”,以使其更具人类可读性。这在一些实施例中可能是有益的,例如当向可能不熟悉针对正则表达式的Unicode表达式的非复杂性正则表达式用户输出时。

[0066] 此外,在一些实施例中,当执行LCS算法时,不是独立地对待每个字符,而是可以将顺序且相等的正则表达式代码转换成跨度数据结构(其也可以被称为跨度)。在一些情况下,跨度可以包括单个正则表达式代码(例如,Unicode广义类别代码)的表示,以及重复计数范围(例如,最小数目和/或最大数目)。从正则表达式代码到跨度的转换可以有助于下面描述的一些不同的附加特征,诸如识别备选(例如,析取),并且还可以有助于合并相邻可选

跨度以进一步简化所生成的正则表达式。

[0067] 如上所述,LCS算法可以被配置为存储和保留输入字符序列内的底层文本片段,其可能被插入回最终的正则表达式中,例如图4中的字符串“iphone”。通过追踪最初引起被分配给该跨度的类别代码的文本片段,这样的实施例可以允许将文字文本(例如,am和pm)直接包括在所生成的正则表达式中,这可以减少假阳性并使正则表达式输出更具人类可读性。

[0068] III.在正则表达式代码组合上使用最长通用子序列算法生成正则表达式

[0069] 本文描述的附加方面涉及基于包括三个或更多个字符串(例如,三个或更多个单独的字符序列)的输入数据来生成正则表达式。当三个或更多个字符串被标识为输入数据时,正则表达式生成器110可以使用性能优化特征,其中针对LCS算法执行的序列确定最佳顺序。如下所述,针对多于两个字符串的性能优化特征可以包括构建图,该图具有对应于每个字符串的顶点,以及可以基于每个字符串和每隔一个字符串之间的LCS输出的大小的边长度/权重。然后,可以使用这些边权重来导出最小生成树,并且可以执行深度优先遍历以确定输入串的顺序。最后,可以使用确定的输入串顺序来完成一系列LCS算法。

[0070] 图5是示出在更大的正则表达式代码集合(例如,三个或更多个字符序列)上使用最长通用子序列(LCS)算法生成正则表达式的过程的流程图。因此,本示例中的步骤502-505可以对应于上面在图3中讨论的步骤303。然而,由于本示例涉及基于三个或更多个输入字符序列生成正则表达式,因此可以多次执行LCS算法。例如,为了避免三个或更多个输入串的运行时间的指数增加,可以多次执行LCS算法,其中每次执行仅在两个输入串上执行。例如,正则表达式生成器110可以对两个字符串(例如,两个输入字符序列或两个转换的正则表达式代码)执行LCS算法的初始执行,然后可以对第一LCS算法的输出和第三字符串执行LCS算法的第二执行,然后可以对第二LCS算法的输出和第四字符串执行LCS算法的第三执行,依此类推。

[0071] 为了改进和/或优化这种实施例的性能,可能希望确定针对输入字符串(例如,输入字符序列或正则表达式代码)的最佳顺序以执行LCS算法序列。例如,获取输入字符串的良好顺序可能会影响所生成的正则表达式的可读性,例如通过最小化可选跨度的数量。为了保持所生成的regex的简明性,LCS到当前regex的附加字符串应优选地已与当前regex(对已经看到的字符串进行LCS得到的中间结果)有些类似。

[0072] 因此,在步骤501中,将多个(例如,3个或更多个)输入字符序列转换为正则表达式代码。在步骤502中,确定使用LCS算法处理正则表达式代码的顺序。下面参照图7更详细地讨论步骤502中的顺序确定。在步骤503中,或者选择所确定顺序的前两个正则表达式代码(对于步骤503的第一次迭代),或者选择所确定顺序的下一个正则表达式代码(对于步骤503的后续迭代)。在步骤504中,对与正则表达式代码格式对应的两个输入字符串执行LCS算法。对于步骤504的第一次迭代,对所确定顺序的前两个正则表达式代码执行LCS算法,而对于步骤504的后续迭代,对所确定顺序的下一个正则表达式代码和前次LCS算法的输出(也可以是相同格式的正则表达式代码)执行LCS算法。在步骤505中,正则表达式生成器110确定所确定的顺序中是否存在尚未作为输入提供给LCS算法的附加正则表达式代码。如果是,则该过程返回到步骤503,用于LCS算法的另一次执行。如果不是,则在步骤506中,基于LCS算法的最后一次执行的输出生成正则表达式。

[0073] 图6是基于五个输入字符序列示例生成正则表达式的示例图。在该示例中,每个输入字符序列被转换为正则表达式代码,然后基于正则表达式代码的确定顺序重复执行LCS算法。因此,图6示出了应用上述图5中讨论的过程的一个示例,在该示例中,五个正则表达式代码的确定顺序是代码#1到代码#5,并且每个代码以所确定顺序输入到LCS算法以生成正则表达式输出。最终正则表达式输出(Reg Ex#4)对应于基于所有五个输入字符序列生成的最终正则表达式。

[0074] 图7是示出用于在更大的正则表达式代码集合(例如,三个或更多)上确定最长通用子序列(LCS)算法的执行顺序的过程的流程图。因此,如本示例所示,步骤701-704可以对应于上面讨论的步骤502中的顺序确定。在步骤701中,可以在对应于输入数据的每个唯一正则表达式代码对上运行LCS算法,并且可以针对每次执行存储所得到的输出LCS。因此,对于k个输入数据,这可以表示要运行LCS算法的所有(k(k-1))/2个可能的字符串配对,或者在一些实施例中表示k(k-1)个。例如,如果接收到k=3个输入字符序列,则在步骤701中可以运行LCS算法三次;如果接收到k=4个输入字符序列,则可以在步骤701中运行LCS算法六次;如果接收到k=5个输入字符序列,则可以在步骤701中运行LCS算法10次,以此类推。在步骤702中,可以由表示字符串的k个节点构成全连接图,其中(k(k-1))/2条边的边权重是两个节点之间的原始LCS输出的长度。在步骤703中,可以从步骤702中的完全连接图中导出最小生成树。在步骤704中,可以对最小生成树执行深度优先遍历。该遍历的输出可以对应于正则表达式代码将被输入到LCS算法执行序列中的顺序。

[0075] 简要参考图8A和8B,图5中示出了基于接收到的k=5个输入字符序列生成的全连接图的示例,并且在图8B中示出了针对全连接图的最小生成树表示。

[0076] 在一些实施例中,图5-8B中描述的方法可以提供关于性能的附加技术优势。例如,LCS算法的某些传统实现可以表现出$O(n^2)$的运行时性能,其中n是字符串的长度。将这种实现扩展到k个字符串而不是仅2个,可能会导致指数运行时间性能$O(n^k)$,因为可能需要LCS算法来搜索k维空间。LCS算法的这种传统实现可能性能不高或不足以适合实时在线用户体验。

[0077] 如上所述,LCS算法可以被执行(k(k-1))/2次,其中有时副本与之前已经看到的完全相同,因为LCS算法可以在来自用户的原始输入示例已经被转换成regex类别代码时。因此,可以在某些情况下实现记忆性,其中可以使用高速缓存来将先前看到的LCS问题映射到先前工作的LCS解决方案。

[0078] IV.基于肯定和否定模式匹配示例的正则表达式生成

[0079] 本文描述的附加方面涉及基于与肯定示例和否定示例两者相对应的输入数据生成正则表达式。如上所述,肯定示例可以指被指定为应与正则表达式生成器将生成的正则表达式匹配的示例字符串的输入数据字符序列。相反,否定示例可以指被指定为不应与正则表达式生成器将生成的正则表达式匹配的示例字符串的输入数据字符序列。如下所述,在一些实施例中,正则表达式生成器110可以被配置为标识将肯定示例和否定示例区分开的位置和该位置处的最短字符子序列。然后,最短子序列可以被硬编码到所生成的正则表达式中,使得肯定示例将与正则表达式匹配,而否定示例将被正则表达式排除(例如,将不匹配)。

[0080] 图9是示出基于肯定和否定字符序列示例生成正则表达式的过程的流程图。在步

骤901中,正则表达式生成器110可以接收对应于肯定示例的一个或多个输入数据字符序列。在步骤902中,正则表达式生成器110可以基于接收到的肯定示例来生成正则表达式。因此,步骤901-902可以包括上面讨论的图3或图5中执行的一些或全部步骤,以基于输入数据字符序列生成正则表达式。

[0081] 在步骤903中,正则表达式生成器110可以接收与否定示例相对应的一个附加输入数据字符序列。因此,特定地指定否定示例,使得与在步骤902中生成的正则表达式不匹配。在一些实施例中,可以首先对照在步骤902中生成的正则表达式来测试在步骤903中接收到的否定示例,并且如果确定该否定示例与正则表达式不匹配,则不采取进一步的动作。然而,在该示例中,可以假设在步骤903中接收的否定示例中的至少一个与在步骤902中生成的正则表达式匹配。因此,在步骤904中,可以在步骤902中生成的正则表达式内确定消歧位置。在一些实施例中,消歧位置可以被选择为前缀位置(例如,在正则表达式的开头)或后缀位置(例如,在正则表达式的末尾)。例如,正则表达式生成器110可以确定在前缀处需要的第一数量的字符以将肯定示例和否定示例区分开,以及在后缀处需要的第二数量的字符以将肯定示例和否定示例区分开。正则表达式生成器110然后可以基于所需的最短数量的替换字符来选择后缀或前缀。在某些情况下,出于可读性的目的,使用前缀作为消歧位置可能是优选的(例如,被加重)。在其他示例中,消歧位置可以是不对应于正则表达式的前缀或后缀的中间跨度位置。

[0082] 在步骤905中,正则表达式生成器110可以确定定制字符类的替换序列,当在所确定的位置处将其插入到正则表达式中时,该替换序列可以区分肯定示例和否定示例。在一些实施例中,正则表达式生成器110在步骤905中可以从与消歧位置(或替换位置)相对应的肯定和否定示例中的每一个检索文本片段,然后使用该文本片段来确定将用作区分肯定示例和否定示例的替换序列的鉴别符。另外,在步骤905中确定的鉴别符替换序列可以包括定制字符类的多个不同的替换序列,其可以在正则表达式内的相同位置或不同位置被替换。

[0083] 如上所述,在一些情况下,可以结合步骤904中的消歧位置(或替换位置)的确定来执行步骤905中的替换序列的确定。例如,正则表达式生成器110可以确定在第一可能的替换位置可以区分肯定和否定示例的一个或多个替换序列。正则表达式生成器110还可以确定在第二不同的可能替换位置可以区分肯定和否定示例的一个或多个其他替换序列。在该示例中,当在不同的可能替换位置和相应的替换序列之间进行选择时,正则表达式生成器110可以应用启发式公式以基于替换位置的字符大小以及相应替换序列的数量和/或大小中的一个或多个来执行选择。最后,在步骤906中,可以通过将一个或多个确定的替换序列插入到所确定位置以替换正则表达式的先前部分来修改正则表达式。在一些情况下,在步骤906中修改正则表达式之后,可以对照修改后的正则表达式来测试肯定示例和/或否定示例,以确认肯定示例匹配并且否定示例不匹配正则表达式。

[0084] 图10A和10B是示出基于肯定和否定字符序列示例生成正则表达式的示例用户界面屏幕。因此,图10A和10B中示出的示例可以对应于在执行上面讨论的图9的过程期间显示的用户界面。在图10A中,用户提供数据输入字符序列1001的三个肯定示例,并且正则表达式生成器110生成与每个肯定示例匹配的正则表达式1002。然后,在图4B中,用户提供一个否定示例1004,并且正则表达式生成器110生成修改的正则表达式1005,其基于肯定示例1003和否定示例1004的两个当前集合。

[0085] 如上所述,在一些实施例中,当接收到肯定示例和否定示例两者时,正则表达式生成器110可以标识区分肯定示例和否定示例的鉴别符或一个或多个字符的最短子序列。所选择的鉴别符可以是最短序列(例如,以类别代码表示),并且既可以是肯定的也可以是否定的,使得肯定示例将匹配而否定示例将不匹配。在一些情况下,鉴别符可以对应于替换子序列,然后可以在步骤905中将该替换子序列硬编码到正则表达式中。作为示例,在"[AL][a-z]+"中,[AL]是肯定鉴别符,假设应用于街道后缀,它将与(或允许)"Alley"、"Avenue"和"Lane"匹配,但不与(或不允许)其他任何词匹配。作为另一示例,在"[BC][o][a-z]+"中,[BC][o]是肯定鉴别符,由两个字符类的序列组成,它们与"Boulevard"和"Court"匹配。作为再一示例,在"[^A][a-z]+"中,[^A]可以是否定鉴别符,其不允许"Alley"和"Avenue"。在某些情况下,该算法以生成否定回溯来正确鉴别。例如,(?<!Av)[A-Za-z]+将排除"Avenue",但允许"Alley"。

[0086] 作为另一示例,如果用户提供了肯定示例"202-456-7800"和"313-678-8900"以及否定示例"404-765-9876"和"515-987-6570",则在某些实施例中,正则表达式生成器110可以生成正则表达式"\d\d\d-\d\d\d-\d\d00"。也就是说,基于确定以00结尾的电话号码区分肯定示例和否定示例(例如,假设目标是匹配商务电话号码的正则表达式),可以针对正则表达式的后缀标识替换字符子序列。这是通过后缀的否定示例的示例(或者更具体地,通过使用肯定后缀容纳否定示例的示例),但是各种其他实施例可以支持前缀、后缀或中间跨度位置的替换。在中间跨度位置替换的示例中,跨度中的字符偏移可以被追踪,并且可以在中间跨度点处拆分。

[0087] 为了决定是使用前缀还是后缀,在一些实施例中,采用试探法,其中在k_a和前缀/后缀的所有组合中选择最小分数:

[0088] $$分数 = k_a \min^2\left\{\frac{|\mathbf{F}_p|}{1+|\mathbf{E}_p|}, \frac{|\mathbf{F}_n|}{1+|\mathbf{E}_n|}\right\} + \begin{cases} 0.0 \text{ 如果是前缀} \\ 0.1 \text{ 如果是后缀} \end{cases}$$

[0089] 其中,

[0090] k_a=考虑用来消歧词缀(前缀或后缀)的字符数

[0091] $|F_p|$=消歧词缀所需的肯定示例中唯一文本片段的数量

[0092] $|F_n|$=消歧词缀所需的否定示例中唯一文本片段的数量

[0093] $|E_p|$=用户提供的(完整)肯定示例数量

[0094] $|E_n|$=用户提供的(完整)否定示例数量

[0095] 在上面的例子中,启发式被设计成支持较短的消歧文本片段而不是较长的文本片段(例如,因此乘以k_a)。启发式也被设计成偏爱前缀而不是后缀(例如,因此针对后缀的惩罚为0.1),以提高可读性。最后,启发式被设计成偏爱对更长的前缀或后缀消歧(例如,替换),而不是通过使用更大数量的字符串片段来消歧(例如,从而是要替换的字符串片段数量的平方)。

[0096] 如上所述,一些实施例还可以支持否定的中间跨度示例以及否定的回溯示例和否定的前瞻示例。

[0097] 一旦确定了前缀/后缀和k(要消歧的字符数),则正则表达式生成器110仍然可以确定如何在所生成的正则表达式中表示该消歧。所生成的正则表达式可以允许看起来像肯定示例的词缀(例如,前缀或后缀),也可以排除看起来像否定示例的词缀。

[0098] $$usePermissive = \frac{|\mathbf{E}_p|}{|\mathbf{F}_p|} - \frac{|\mathbf{E}_n|}{|\mathbf{F}_n|}$$

[0099] 如果usePermissive大于零,则通过生成允许从肯定示例中逐个(针对每个字符位置)获取字符的正则表达式来允许看起来像肯定示例的内容通过。在其他情况下,正则表达式生成器110可以通过生成不允许从否定示例中逐个(针对每个字符位置)获取字符的正则表达式来采取不允许看起来像否定示例的内容的方法。

[0100] 作为另一示例,针对肯定示例8am和否定示例9pm生成的正则表达式可以是\d[^p]m。这使用插入符号语法。在一些情况下,正则表达式生成器110可以被配置为偏爱较短的正则表达式,其不仅对用户更具可读性,而且更有可能是正确的。理由是,频繁出现的字符在未来更有可能再次出现,因此应该把重点放在频繁出现的字符上。如果唯一字符$|F_p|$较少(唯一字符较少,因为出现的字符出现的频率更高),则这在启发式中的奖励是将其放在分母中。

[0101] 再次参考上面的usePermissive示例启发式方法,如果只有一个来自用户的肯定示例,则确定一个唯一的肯定词缀并不是什么了不起的事情。因此,在该启发式方法中,低$|E_p|$通过将其置于分子来惩罚(即,在该启发式方法中奖励高$|E_p|$)。

[0102] 另外,在一些实施例中,否定示例可以基于回溯和/或前瞻。例如,用户可以提供“323-1234”的肯定示例和“202-754-9876”的否定示例,则这涉及使用regex回溯语法(?<!)以排除带区号的电话号码。

[0103] 在某些情况下,否定示例也可以基于可选跨度。例如,用户可以提供“ab”和“a2b”的肯定示例以及“a3b”的否定示例。在这种情况下,示例实现可能会失败,因为它可能尝试仅基于所需的跨度进行区分,并且“2”数字在可选跨度中。在本示例中,失败可能指的是所生成的正则表达式与所有肯定示例匹配(正确),也与一个或多个否定示例匹配(错误)的情况。在这种情况下,可以向用户警告该故障,并且可以经由用户界面向用户提供手动修复所生成的正则表达式和/或移除一些否定示例的选项。

[0104] V.用于正则表达式生成的用户界面

[0105] 本文描述的附加方面包括图形用户界面内与正则表达式生成相关的若干不同特征和功能。如下所述,这些特征中的某些特征可以包括用于用户选择和突出显示肯定和否定示例、针对肯定和否定示例的颜色编码、以及数据单元格内的多个重叠/嵌套突出显示的各种选项。

[0106] 图11是示出基于在用户界面内接收的用户数据选择来生成正则表达式的过程的流程图。图11中的示例过程可以对应于基于输入数据字符序列生成正则表达式的任何前面讨论的示例。然而,图11描述了关于可以在客户端设备120上生成和显示的用户界面的过程。在步骤1101中,响应于来自用户经由用户界面的请求,正则表达式生成器110可以(例如,从数据存储库130)检索数据,并在图形用户界面内以表格形式呈现/显示数据。尽管在这个示例中使用了表格数据,但是应该理解,在其他示例中不需要使用或显示表格数据。例如,在某些情况下,用户可以直接键入原始数据(而不是从用户界面选择数据)。另外,当经由用户界面呈现数据时,数据不需要以表格形式,而可以是非结构化数据(例如,文档)或半结构化(例如,诸如推文或帖子的非格式化/非结构化数据项的电子表格)。在各种示例中,表格数据可以对应于交易数据、客户数据、绩效数据、预测数据和/或可以存储在企业或其

他组织的数据存储库130中的任何其他类别的数据。在步骤1102中,可以经由用户界面接收用户对输入数据的选择。例如,所选择的输入数据可以对应于由用户选择的整个数据单元,或者数据单元内的字符的子序列。在步骤1103中,正则表达式生成器110可以基于在步骤1102中接收的输入数据(例如,数据单元或其部分)来生成正则表达式。在步骤1104中,可以响应于正则表达式的生成来更新用户界面。在一些情况下,可以简单地更新用户界面以向用户显示所生成的正则表达式,而在其他情况下,用户界面可以按照下面讨论的各种其他方式来更新。如本示例所示,用户可以经由用户界面选择多个不同的输入数据字符序列,并且响应于接收到的每个新的输入数据,正则表达式生成器110可以生成包含字符序列的第一和第二(肯定)示例的更新的正则表达式。然后,当用户突出显示第三字符序列时(例如,在两个字符序列之外,或者在第一或第二字符序列内),正则表达式生成器110可以再次更新正则表达式,依此类推。在一些实施例中,正则表达式生成器110可以实时(或接近实时)执行该算法,使得响应于用户进行的每一个新的击键或每一个新的突出显示部分来生成全新的正则表达式。

[0107] 因此,如图11所示,响应于用户经由用户界面对字符序列的选择,正则表达式生成器110可以生成并显示正则表达式。例如,当用户突出显示第一字符序列时,正则表达式生成器可以生成并显示表示第一字符序列的正则表达式。当用户突出显示第二字符序列时,正则表达式生成器可以生成包含第一和第二字符序列的更新的正则表达式。然后,当用户突出显示第三字符序列(例如,在第一或第二序列内)时,正则表达式生成器可以再次更新正则表达式,以此类推。

[0108] 图12是示出经由在用户界面内接收的用户数据选择,基于捕获组来生成正则表达式和提取数据的过程的另一流程图。在步骤1201中,如上文在步骤1101中所讨论的,正则表达式生成器110可以(例如,从数据存储库130)检索数据,并在图形用户界面内以表格形式呈现/显示数据。在步骤1202中,正则表达式生成器110可以接收对特定数据单元格内的文本片段的用户突出显示的选择。在步骤1203中,正则表达式生成器110可以基于所选数据单元格的肯定示例来生成正则表达式,并且在步骤1204中可以基于单元格内突出显示的文本片段来创建正则表达式捕获组。在步骤1205中,正则表达式生成器110可以确定所显示的表格数据中与所生成的正则表达式相匹配的一个或多个附加单元格,并且在步骤1206中,可以提取与所生成的正则表达式相匹配的附加单元格中的相应文本片段。

[0109] 因此,除了提供肯定示例之外,用户还可以选择(例如,经由鼠标文本突出显示)任何所选肯定示例内的文本片段。作为响应,正则表达式生成器110可以创建正则表达式捕获组以从该示例中提取该文本片段以及从该正则表达式应用到的文本中的所有其他匹配中提取相应片段。从匹配的数据单元中提取文本片段还可以包括删除和修改,并且在某些情况下可以用于在半结构化或非结构化文本的现有列之外创建新的数据列。

[0110] 使用用户选择肯定数据示例的示例,并且如果用户突出显示年份,则正则表达式生成器110可以生成正则表达式(?:[A-Z]{3}\s+\d\d,\s+|\d\d/\d\d)(\d\d\d\d)。如本示例所示,正则表达式生成器110已经在年份周围放置了圆括号,并且还通过使用?:正则表达式语法将围绕月和日(用于备选)的旧圆括号转换为“非捕获”组。在一些实施例中,可能需要提取/捕获组落在跨度边界上,并且在这样的实施例中,正则表达式生成器110可以将突出显示的字符范围作为输入并将其扩展以包含最近的锚跨度边界。然而,在其他示例中,用

户界面可以支持中间跨度提取/捕获。

[0111] 在一些实施例中,用户界面可以支持来自用户的包括对第二字符序列内的第一字符序列的选择的输入数据。例如,用户可以突出显示较大的先前突出显示的字符序列内的一个或多个字符,并且第二用户选择可以针对较大的第一用户选择提供上下文。这样的实施例可以使输入数据能够更有针对性地提供给正则表达式生成器110。

[0112] 此外,在一些示例中,可以响应于用户在用户界面进行选择(例如,突出显示文本)来发起操作并且可以打开对话框。在一些情况下,该对话框可以是非模型对话框,例如不阻止用户与主屏幕交互的浮动工具箱窗口。取决于用户正在执行的主要操作,该对话框还可以在外观和/或功能上改变。因此,在这种情况下,用户不需要在突出显示所选文本之后搜索其他菜单项,以便启动对捕获组文本片段的修改、提取等。此外,在某些实施例中,提供用于生成正则表达式的用户界面可以包括三种突出显示模式:嵌套自动、嵌套手动和单层。在某些情况下,默认操作模式可以是将整个单元格标识为突出显示区域,并且用户还可以突出显示所突出显示单元格内的一个或多个附加子序列。在其他模式中,可以允许用户在表格数据显示的数据单元格内手动指定两个突出显示。在其他模式中,可以允许用户手动指定外部突出显示,而没有内部突出显示。这些其他模式可能更适合于“半结构化”数据,例如,由推文或其他长字符串(例如浏览器“用户代理”字符串)组成的数据列。“半结构化”数据指的是可以在用户界面内以表格形式显示的数据,但是该表格内的列由非结构化文本组成。

[0113] 在一些这样的实施例中,用户经由用户界面进行的内部和外部选择(例如,突出显示)可以通过颜色编码来区分。例如,肯定示例的外部突出显示可以以第一文本/背景颜色组合显示,而肯定示例的内部突出显示可以以不同的对比文本/背景颜色组合显示。

[0114] 如上所述,用户可以经由选择字符子序列来指定捕获组的选择。该GUI可用于经由突出显示(或其他指示)方便用户选择。图13示出了示例,其中示出了具有表格数据显示的示例用户界面屏幕。在该示例中,图13描绘了例如由用户在列值的一个或多个所需元素上拖拽鼠标引起的列值内的突出显示。注意,在其中执行用户突出显示的“单元格”可以表现出指示选择列值的颜色改变。该颜色改变可以被解释为响应于用户突出显示而自动突出显示。

[0115] 图14和15是示例用户界面屏幕,示出了基于从表格显示中选择数据来生成正则表达式和捕获组。在这些示例中,图14和图15示出了附加用户界面窗口,其自动显示检测到表格数据显示内的用户突出显示1401。该窗口包括用于显示肯定示例的字段1402、用于显示否定示例的字段、以及用于响应于从表格数据显示中选择肯定示例而显示动态(并且几乎即时地)生成的正则表达式的字段。在这些示例中,列值1401内的用户突出显示可以等同于自动突出显示内的用户突出显示。因此,用户突出显示区号不仅导致用户突出显示的区号1401,而且还导致电话号码的其余部分被填充在肯定示例字段1402中。

[0116] 然而,应该理解,用户突出显示不限于自动突出显示内的性能。例如,备选地,可以在其他用户突出显示内执行用户突出显示。作为另一示例,用户突出显示可以备选地在没有任何内部突出显示的情况下执行(例如,在突出显示的文本内进一步突出显示)。这些备选的示例特别适合于半结构化数据,例如包括“Tweets”或其他长字符串(例如,浏览器“用户代理”字符串)的数据列。

[0117] 此外,在生成相应的正则表达式时,可以基于附加的自动突出显示来标识与该正则表达式匹配的其他列值1402。在图14和15所示的示例中,附加的自动突出显示指示与所生成的正则表达式的捕获组相匹配的这些其他列值的元素。可以使用与用于用户突出显示的颜色不同的颜色来执行附加的自动突出显示。

[0118] 如图15所示,示出了附加的用户突出显示以指示用户选择其他示例。可以以与上述类似的方式执行附加的用户突出显示。因此,图15中的用户界面示出了用于显示肯定示例的字段1502中的其他示例的填充。这可以响应于检测到附加的用户突出显示而发生。另外,所生成的正则表达式1503可以被动态地并且几乎即时地更新,使得它与所有肯定示例1502匹配。响应于生成更新的正则表达式,也可以更新与更新的正则表达式匹配的其他列值1504的自动突出显示。在一些实现中,还可以使用动态颜色编码。例如,可以使用第一颜色(例如,蓝色)对匹配进行颜色编码,而使用第二颜色(例如,绿色)对肯定示例进行颜色编码,并且可以使用第三颜色(例如,红色)对否定示例进行颜色编码。

[0119] 图16A和16B是示例用户界面屏幕,示出了基于从表格显示中选择肯定示例和否定示例来生成正则表达式。在图16A-16B中,来自肯定示例字段1602的个别示例可以从肯定示例字段1603中移除,和/或被移动到否定示例字段1603。在用户界面内,这可以例如通过用户在示例之一上点击(例如,右击)以选择它来执行。该选择可以使用户界面显示包括删除选项和改变选项的菜单1602。此后,在选项上点击可以导致相应功能的执行。

[0120] 在图16A和16B所示的示例中,用户选择改变选项的结果是将所选示例移动到否定示例字段1603,使得正则表达式1601被更新为正则表达式1604,该正则表达式1604可以动态地和近乎即时地(例如,在某些实施例中在30ms和9000ms之间)生成。响应于生成更新的正则表达式1604,与更新的正则表达式匹配的其他列值的自动突出显示也可以在表格数据显示内更新。此外,可以对一些或全部否定示例执行自动突出显示,包括对应于否定示例的任何列值,其可以使用与上面使用的任何颜色不同的颜色来突出显示,或者使用其他视觉技术在用户界面内以其他方式区分。

[0121] 在一些实施例中,经由用户界面指定否定示例不需要首先将该示例指定为肯定示例,然后将其转换为否定示例,如图16A和16B所示。相反,否定示例可以以多种方式来指定。例如,用户可以经由用户界面选择(例如,右击)列值(例如,对其执行自动突出显示以指示其与所生成的正则表达式匹配的其他列值之一),从而可以导致显示包括将所选列值指定为否定示例的选项(例如,“创建新的反例”)的菜单。

[0122] 因此,使用图16A和16B中所示的示例,响应于生成更新的正则表达式1604,也可以更新与更新的正则表达式匹配的其他列值的自动突出显示。在这些示例中,更新的正则表达式指定以“9”结尾的电话号码。

[0123] 简要返回到图14和15,当用户点击或以其他方式选择“提取”按钮时,可以启动操作以提取与当前正则表达式1403或1503匹配的所有单元格内的突出显示的文本片段。尽管在图14和图15中未示出,但在一些实施例中,用户界面可以附加于或替代“提取”按钮提供其他可选择的按钮。例如,“替换”按钮可以被呈现作为选项以由用户指定的元素替换用户突出显示的元素。另外地或备选地,一个或多个“删除”按钮可以被呈现为选项,以实际上用空来替换用户突出显示的元素。例如,可以实现“删除片段”操作和/或“删除行”操作中的一个或两者,这将分别删除用户突出显示的文本片段或行。可以在各种实施例中实现的附加

操作可以包括:“保留行”操作、“分割”操作(例如,突出显示逗号,然后将逗号分隔的组分提取到分开的多个新列中),以及“模糊”操作(例如,用“#”序列或其他符号替换突出显示的文本/捕获组)。

[0124] 在该示例中,响应于选择“提取”按钮,提取操作可以被添加到要由下游操作执行的转换脚本列表。在一些实施例中,转换脚本列表可以被显示在用户界面的一部分中以供用户查看/修改。备选地,可以在现场执行提取操作以生成包括正则表达式捕获组的内容的新列(例如,与肯定示例的用户突出显示的部分相对应的元素)。在图14和15所示的示例中,可以响应于对“提取”按钮的选择而生成区号的新列和/或新表格。

[0125] 图17是根据本文描述的一个或多个实施例的另一示例用户界面屏幕,其示出了基于从表格显示中选择数据来生成正则表达式和捕获组。

[0126] VI.在跨度上使用最长通用子序列算法的正则表达式生成

[0127] 本文描述的附加方面涉及基于来自一个或多个数据输入字符序列的LCS算法的正则表达式生成,但是其中正则表达式生成器110还可以处理仅在一些示例中出现的字符。为了处理仅在一些输入示例中出现的字符,可以定义跨度,其中追踪正则表达式代码的最小和最大出现次数。例如,对于“9pm”和“9pm”的字符序列输入,在数字和“pm”文本之间存在可选的空格。在这种情况下,当在所有给定输入示例中可能不存在某一跨度(例如,“9”和“pm”之间的单个空格)时,最小出现次数可以被设置为零。然后可以将这些最小和最大数字映射到正则表达式重数语法。最长通用子序列(LCS)算法可以在从输入示例得到的字符的跨度上运行,包括并非在每个输入示例中都出现的“可选”跨度(例如,最小长度为零)。如下所述,可以在LCS算法的执行期间合并连续跨度。在这种情况下,当携带的额外可选跨度不再连续出现时,LCS算法也可以在这些可选跨度上递归运行。也就是说,尽管LCS算法的运行本质上是递归的,但在这些情况下,整个LCS算法可以递归运行(例如,递归运行递归LCS算法)。在其他技术优势中,这可以允许更短、更整洁、更具可读性的正则表达式生成。例如,可以在不递归运行LCS算法的情况下生成(am|am)(即,在am之前带有可选空格),而递归运行LCS算法可以导致正则表达式被生成为(?am),它更短并且更整洁。

[0128] 图18是根据本文描述的一个或多个实施例,示出使用最长通用子序列(LCS)算法生成包括可选跨度的正则表达式的过程的流程图。在步骤1801中,正则表达式生成器110可以接收对应于肯定正则表达式示例的一个或多个字符序列作为输入数据。在步骤1802中,正则表达式生成器110可以将字符序列转换为正则表达式代码。因此,步骤1801和1802可以与上面讨论的前面相应示例类似或相同。然后,在步骤1802中,还可以将正则表达式代码转换成跨度数据结构(或跨度)。如上所述,每个跨度可以包括存储字符类代码(例如,regex代码)和重复计数范围(例如,最小计数和/或最大计数)的数据结构。在步骤1804中,正则表达式生成器110可以执行LCS算法,将跨度集合作为输入提供给该算法。该示例中的LCS算法的输出可以包括跨度的输出集合,包括具有等于零的最小重复计数范围的至少一个跨度,其对应于LCS算法的输出内的可选跨度。最后,在步骤1805中,正则表达式生成器110可以基于LCS算法的输出(包括可选跨度)来生成正则表达式。

[0129] 图19是示出使用最长通用子序列(LCS)算法生成正则表达式的示例图,其中所生成的正则表达式包括可选跨度。在该示例中,两个输入数据字符序列是“8am”和“9pm”。如上所述,输入数据字符序列首先被转换成正则表达式代码(步骤1802),然后被转换成跨度(步

骤1803)。跨度可以作为输入提供给LCS算法(步骤1804),并且LCS输出包括可选跨度$Z^{\langle 0,1\rangle}$,指示可选的单个空格可以是数字和两个字母的文本序列。也就是说,本示例中的上标记法可以包括两个数字,即应用于在前代码(例如,Z=空格)的最小重复计数范围(例如,0)和最大重复计数范围(例如,1)。最后,可以基于LCS算法的输出跨度来生成正则表达式,并且可以将可选跨度转换为相应的正则表达式代码“pZ^{*}”。

[0130] 在一些实施例中,正则表达式生成器110在LCS算法的执行期间再现和使用可选空间可以在性能和可读性方面提供额外的技术优势。例如,当生成正则表达式时,在某些情况下希望能够处理所有给定示例中通用的字符和仅在一些示例中出现的字符。

[0131] 在某些实施例中,对于每个跨度数据结构,可以追踪类别代码的最小出现次数和类别代码的最大出现次数。在一个或多个给定示例中根本不存在跨度的情况下,将最小数字设置为零。作为另一示例,为了生成正则表达式来处理拼写出的一年中的月份,最小和最大数字然后可以被映射到包括花括号的正则表达式重数语法(例如,[A-Za-z]{3,9})。

[0132] 在一些实施例中,正则表达式生成器110可以追踪针对每个跨度的最小和最大出现次数,但也可以处理附加的实现细节。例如,作为处理可选跨度和在字符跨度上运行LCS的组合的结果,正则表达式生成器110可以被配置成在LCS算法的整个执行过程中检测和合并连续跨度。此外,所携带的任何额外的可选跨度有时会连续出现,并且LCS算法也可能需要在这些跨度上递归运行。例如,在一些情况下,正则表达式生成器110修改和/或扩展LCS算法以倾向(或加权)可选和所需序列元素(例如,跨度)之间的较少转换。例如,将可选跨度分组在一起可以最小化必须在正则表达式中使用的分组圆括号的数量,从而可以提高所生成的正则表达式的人类可读性。在一些情况下,如果即使在考虑了可选跨度之后,所得到的长度也相等,则正则表达式生成器110可以表现出对可选和所需跨度之间具有较少转换的备选方案的偏好。例如,在某些情况下,可以实现标准的LCS算法以偏好在其判定点处选择较长的序列。然而,在选项具有相等长度的判定点,可以将配置偏好编程到正则表达式生成器110中。例如,一种这样的配置偏好可以是偏好更短的序列(一旦考虑到可选跨度)。因此,该配置内的定制LCS可以同时针对(所需跨度的)较长序列和(总所需跨度和可选跨度)的较短序列进行优化。

[0133] 在一些实施例中,如果所生成的正则表达式以所需跨度开始(这也可以用作对人类读者的心理锚),而不是以可选跨度开始正则表达式,则所生成的正则表达式可能更具可读性。因此,在一些情况下,如果所得到的选项具有相等数量的转换,则可以选择具有较早非可选跨度的选项。另外,在一些实施例中,正则表达式生成器110执行的LCS算法可以被配置为将所有空格(包括对应于空格的可选跨度)推到正则表达式内的右侧。通过将所有空格推到右侧,可能会增加将空格跨度合并在一起的可能性,这可以简化所得到的正则表达式,并且提高可读性。因此,在LCS算法的执行期间,当确定两个子串集合具有相同的LCS时,可以选择有助于提高可读性的集合,而不是任意选择这两个集合中的一个。此外,在一些实施例中,LCS算法可以被配置为倾向更大数量的所需跨度和/或更少的可选跨度,以便提高可读性。

[0134] 如上所述,在某些情况下,否定示例也可以基于可选跨度。例如,用户可以提供“ab”和“a2b”的肯定示例以及“a3b”的否定示例。在这种情况下,示例实现可能会失败,因为它可能尝试仅基于所需跨度进行鉴别,并且“2”数字在可选跨度中。在这种情况下,可以向

用户警告该失败,并且可以经由用户界面向用户提供选项以手动修复所生成的正则表达式和/或移除一些否定示例。

[0135] 在一些实施例中,可以存在作为从REST服务返回的JSON的一部分返回的isSuccess。在一些实施例中,当isSuccess=false时,所生成的regex可以变成不同的颜色(例如,红色)。

[0136] VII.使用组合最长通用子序列算法的正则表达式生成

[0137] 本文描述的其他方面涉及组合搜索,其中由正则表达式生成器110执行的LCS算法可以被多次运行,以生成"正确的"正则表达式(例如,与所有给定的肯定示例适当匹配并且适当地排除所有给定的否定示例的正则表达式),和/或生成多个正确的正则表达式,从中可以选择最期望的或最佳的正则表达式。例如,在组合搜索期间,完整的LCS算法和正则表达式生成过程可以运行多次,包括文本处理方向的不同组合/排列、不同的锚定以及LCS算法的其他不同特性。

[0138] 图20是示出基于最长通用子序列(LCS)算法的组合执行来生成正则表达式的过程的流程图。在步骤2001中,正则表达式生成器110可以接收与肯定示例相对应的输入数据字符序列。在步骤2002中,正则表达式生成器110可以迭代LCS算法的执行技术的各种不同组合。如该示例所示,在步骤2002的每次迭代期间,正则表达式生成器110可以选择以下LCS算法执行参数(或特性)的不同组合:锚定(即,不锚定、锚定到行的开头、锚定到行的末尾)、处理方向(即,从右到左的顺序、从左到右的顺序)、推入空格(即,推入或不推入空格)和折叠跨度(即,折叠或不折叠跨度)。在步骤2003中,在输入数据字符序列上运行LCS算法(或者,如果输入字符序列首先被转换,则在正则表达式代码上运行),其中基于在步骤2002中选择的参数/特性来配置LCS算法。在步骤2004中,LCS算法的输出可以由正则表达式生成器110存储,包括诸如该算法是否成功标识了LCS以及相应的正则表达式的长度的数据。在步骤2005中,该过程可以迭代,直到LCS算法已经用组合搜索的参数/特性的所有可能组合运行。最后,在步骤2006中,来自LCS之一的特定输出被选择作为最佳输出(例如,基于成功和正则表达式长度),并且可以基于所选择的LCS算法输出来生成正则表达式。

[0139] 在各种实施例中,可以针对参数/特性的各种不同组合执行诸如以上参考图20描述的组合搜索。例如,在一些实施例中,LCS算法可以使用插入符号^将正则表达式锚定到文本的开头,和/或使用美元符号$将正则表达式锚定到文本的末尾。在某些情况下,这种锚定可能会导致生成较短的正则表达式。当用户希望在字符串的开头和/或结尾处找到特定模式时,锚可能特别有用。例如,用户可能在开头想要产品名称。为了避免将LCS算法与描述产品名称的不同数量的单词混淆,可以使用插入符号将regex锚定到字符串的开头,如下图所示。

[0140] 此外,在一些实施例中,可以利用正向或反向的输入数据来执行LCS算法(或者类似地,LCS算法可以被配置为以通常的顺序接收输入数据,然后在执行该算法之前反转该顺序)。因此,在一些实施例中,可以对输入字符序列或代码对执行的LCS算法的组合搜索可以是:

[0141] 1.通常(从右到左)顺序,不锚定到开始或结束

[0142] 2.通常(从右到左)顺序,使用插入符号^锚定到行首

[0143] 3.通常(从右到左)顺序,使用美元$锚定到行尾

[0144] 4.颠倒(从左至右)顺序,不锚定到开始或结束

[0145] 5.颠倒(从左到右)顺序,使用插入符号^锚定到行首

[0146] 6.颠倒(从左到右)顺序,使用美元$锚定到行尾

[0147] 在该示例中,在LCS的六次执行中,可以选择最短的所得到的正则表达式(步骤2006)。

[0148] 在一些实施例中,LCS算法的组合搜索还可以在贪婪量词“?”和非贪婪的量词“??”上迭代。例如,默认情况下,如果存在可选跨度,则会发出单个问号,例如[A-Z]+(?:[A-Z]\.)?[A-Z]+表示名和姓,中间首字母可选。如果在使用贪婪量词时找不到令人满意的正则表达式,则组合搜索可以尝试将所有问号量词替换为双问号量词(例如,[A-Z]+(?:[A-Z]\.)??[A-Z]+)。双问号对应于非贪婪量词,其可以指示下游正则表达式匹配器进入回溯模式以便找到匹配。

[0149] 此外,在一些实施例中,LCS算法的组合搜索还可以迭代是否优选右侧的空格。例如,如上所述,可以在一些实施例中使用将空格推到右侧的策略,例如,当LCS算法面临以其他方式相等的选项的任意选择时,希望空格跨度可以合并在一起,从而产生较少的总体跨度。这个特征向组合搜索添加了另一选项,即要么将空格推到右侧,要么按照传统的LCS方法执行,即任意地做出判定。

[0150] 此外,在一些实施例中,LCS算法的组合搜索还可以通过对原始字符串运行LCS来遍历扫描/不扫描所有示例中通用的文字。在这样的实施例中,LCS算法可以被配置为标识并对齐通用单词。如本文所使用的,“通用词”可以指出现在每个肯定示例中的单词。一旦标识出通用单词,就可以将其跨度类型从字母转换为单词,然后通过LCS算法的后续运行可以自然地将其对齐。

[0151] 因此,在下面的示例中,组合搜索可以遍历几个参数/特性以达到执行完整LCS算法96次。本示例中要遍历的各种参数/特性是:

[0152] • 锚定(3)(值=^、$或两者都不是)

[0153] • 推动空格(2)(值=是或否)

[0154] • 合并低基数跨度到通配符(2)(值=是或否)

[0155] • 贪婪量词?(2)(值=是或否)

[0156] • 在通用令牌上对齐LCS算法(2)(值=是或否)

[0157] • 使用“\w”表示字母数字,还是将字母“\pL”和数字“\pN”视为单独的跨度(2)(值=是或否)

[0158] 如上所述,在该示例中,完整的LCS算法将被执行96次(例如,3*2*2*2*2*2=96)。

[0159] 然而,在其他实施例中,正则表达式生成器110可以提供性能增强,通过该性能增强,仅上述列表中的前三个特性(锚定、推动空格以及将低基数跨度合并为通配符)可以参与组合搜索。这可能导致要执行的完整LCS算法的次数少得多(例如,3*2*2=12次)。在这样的实施例中,尽管上述列表中的最后三个特性(贪婪量词、将LCS算法对齐到通用令牌、以及使用“\w”来表示字母数字,还是将字母“\pL”和数字“\pN”视为单独的跨度)不参与组合搜索,但是这些特性可以在结束时被单独地和顺序地测试。在这样的实施例中可以实现技术优势,因为以这种方式划分搜索空间仍然可以找到令人满意的正则表达式,但是具有大约8倍的性能加速。

[0160] 为了说明这一点,下面的组合搜索示例可以提供优于先前示例的性能优势。在该示例中,组合搜索可以基于要遍历的以下参数/特性来执行:

[0161] • 锚定(3):BEGINNING_OF_LINE_MODE、END_OF_LINE_MODE、NO_EOL_MODE

[0162] • 顺序/方向(2):从右至左(正常)LCS与从左至右(反向)LCS

[0163] • 推动(2):是否尝试在LCS算法内将空格推到右侧

[0164] • 压缩为通配符(2):是否尝试将仅有时出现的长序列向下压缩为通配符.*?

[0165] 本示例中的组合可以导致运行完整算法3*2*2*2=24次。然后,正则表达式生成器110可以取LCS算法的24个结果中的最好结果,其中"最好"可以意味着(a)LCS算法成功,以及(b)生成了最短的正则表达式。正则表达式生成器110然后可以执行以下三个附加任务:

[0166] 1.尝试将不被空格、标点符号或符号间断的字母和数字序列压缩为新的跨度类型I,称为字母数字,对应于所生成的正则表达式\w。这对于在点击流日志的IPv6地址中找到的十六进制数字可能很有用(请参阅2019年4月的新颖性64)。

[0167] 2.尝试使用非贪婪量词??而不是贪婪量词?

[0168] 3.尝试在文字上对齐

[0169] VIII.硬件概述

[0170] 图21描绘了用于实现实施例的分布式系统2100的简化图。在所示实施例中,分布式系统2100包括经由一个或多个通信网络2110耦合到服务器2112的一个或多个客户端计算设备2102、2104、2106和2108。客户端计算设备2102、2104、2106和2108可以被配置为执行一个或多个应用。

[0171] 在各种实施例中,服务器2112可以适合于运行一个或多个服务或软件应用,这些服务或软件应用使得能够自动生成正则表达式,如本公开中所描述的。例如,在某些实施例中,服务器2112可以接收从客户端设备发送的用户输入数据,其中用户输入数据由客户端设备通过在客户端设备处显示的用户界面来接收。服务器2112然后可以将用户输入数据转换成正则表达式,该正则表达式被传输到客户端设备以通过用户界面显示。

[0172] 在某些实施例中,服务器2112还可以提供可以包括非虚拟和虚拟环境的其他服务或软件应用。在一些实施例中,这些服务可以作为基于web的服务或云服务,例如在软件即服务(SaaS)模型下,提供给客户端计算设备2102、2104、2106和/或2108的用户。操作客户端计算设备2102、2104、2106和/或2108的用户进而可以利用一个或多个客户端应用来与服务器2112交互,以利用由这些组件提供的服务。

[0173] 在图21所示的配置中,服务器2112可以包括实现由服务器2112执行的功能的一个或多个组件2118、2120和2122。这些组件可以包括可由一个或多个处理器执行的软件组件、硬件组件或其组合。应当理解,各种不同的系统配置是可能的,其可以不同于分布式系统2100。因此,图21所示的实施例是用于实现实施例系统的分布式系统的一个示例,并且不意图是限制性的。

[0174] 用户可以使用客户端计算设备2102、2104、2106和/或2108来执行一个或多个应用,这些应用可以根据本公开的教导生成正则表达式。客户端设备可以提供使得客户端设备的用户能够与客户端设备交互的接口。客户端设备还可以经由该接口向用户输出信息。尽管图21仅描绘了四个客户端计算设备,但是可以支持任意数量的客户端计算设备。

[0175] 客户端设备可以包括各种类型的计算系统,例如便携式手持设备、诸如个人计算

机和膝上型计算机的通用计算机、工作站计算机、可穿戴设备、游戏系统、瘦客户端、各种消息收发设备、传感器或其他传感设备等。这些计算设备可以运行各种类型和版本的软件应用和操作系统(例如,微软Windows®、苹果Macintosh®、UNIX®或类UNIX操作系统、诸如Google Chrome™OS的LINUX或类LINUX操作系统),包括各种移动操作系统(例如,微软Windows Mobile®、iOS®、Windows Phone®、Android™、BlackBerry®、PalmOS®)。便携式手持设备可以包括蜂窝电话、智能电话(例如,iPhone®)、平板电脑(例如,iPad®)、个人数字助理(PDA)等。可穿戴设备可以包括Google Glass®头盔显示器和其他设备。游戏系统可以包括各种手持游戏设备、支持互联网的游戏设备(例如,具有或不具有Kinect®手势输入设备的Microsoft Xbox®游戏控制台、Sony PlayStation®系统、由Nintendo®提供的各种游戏系统等)等等。客户端设备能够执行各种不同的应用,例如各种与互联网相关的应用、通信应用(例如,电子邮件应用、短消息服务(SMS)应用),并且可以使用各种通信协议。

[0176] 网络2110可以是本领域技术人员熟悉的任何类型的网络,其可以使用各种可用协议中的任何一种来支持数据通信,这些协议包括但不限于,TCP/IP(传输控制协议/互联网协议)、SNA(系统网络体系结构)、IPX(互联网分组交换)、AppleTalk®等。仅作为示例,网络2110可以是局域网(LAN)、基于以太网的网络、令牌环、广域网(WAN)、互联网、虚拟网、虚拟专用网(VPN)、内联网、外联网、公共交换电话网(PSTN)、红外线网络、无线网络(例如,在电气和电子协会(IEEE)1002.11协议组中的任一个、Bluetooth®和/或任何其他无线协议下操作的网络)。

[0177] 服务器2112可以由一个或多个通用计算机、专用服务器计算机(例如包括PC(个人计算机)服务器、UNIX®服务器、中型服务器、大型计算机、机架式服务器等)、服务器群、服务器集群或任何其他适当的布置和/或组合组成。服务器2112可以包括运行虚拟操作系统的一个或多个虚拟机,或者涉及虚拟化的其他计算体系结构,诸如可以被虚拟化以维护服务器的虚拟存储设备的一个或多个灵活的逻辑存储设备池。在各种实施例中,服务器2112可以适合于运行提供前述公开中描述的功能的一个或多个服务或软件应用。

[0178] 服务器2112中的计算系统可以运行一个或多个操作系统,包括上面讨论的操作系统中的任何一个,以及任何市场上可买到的服务器操作系统。服务器2112还可以运行各种附加服务器应用和/或中间层应用中的任何一个,包括HTTP(超文本传输协议)服务器、FTP(文件传输协议)服务器、CGI(通用网关接口)服务器、JAVA®服务器、数据库服务器等。示例性数据库服务器包括但不限于可从

Oracle®、Microsoft®、Sybase®、IBM®(International Business Machines)等商购到的数据库服务器。

[0179] 在一些实现中,服务器2112可以包括一个或多个应用,以分析和合并从客户端计算设备2102、2104、2106和2108的用户接收的数据馈送和/或事件更新。作为示例,数据馈送和/或事件更新可以包括但不限于Twitter®馈送、Facebook®更新或从一个或多个第

三方信息源接收的实时更新和连续数据流,其可以包括与传感器数据应用、金融自动收报机、网络性能测量工具(例如,网络监控和流量管理应用)、点击流分析工具、汽车流量监控等相关的实时事件。服务器2112还可以包括用于经由客户端计算设备2102、2104、2106和2108的一个或多个显示设备显示数据馈送和/或实时事件的一个或多个应用。

[0180] 分布式系统2100还可以包括一个或多个数据存储库2114、2116。在某些实施例中,这些数据存储库可以用于存储数据和其他信息。例如,数据存储库2114、2116中的一个或多个可用于存储信息,例如与系统生成的正则表达式匹配的新数据列。数据存储库2114、2116可以驻留在各种位置。例如,服务器2112使用的数据存储库可以在服务器2112本地或者可以远离服务器2112并且经由基于网络的或专用连接与服务器2112通信。数据存储库2114、2116可以是不同类型的。在某些实施例中,服务器2112使用的数据存储库可以是数据库,例如关系数据库,例如由Oracle Corporation®和其他供应商提供的数据库。这些数据库中的一个或多个可以适于响应于SQL格式的命令向和从数据库存储、更新以及检索数据。

[0181] 在某些实施例中,数据存储库2114、2116中的一个或多个也可以由应用用来存储应用数据。应用使用的数据存储库可以是不同类型的,例如由文件系统支持的键值存储库、对象存储库或通用存储库。

[0182] 在某些实施例中,本公开中描述的功能可以经由云环境作为服务来提供。图22是根据某些示例的基于云的系统环境的简化框图,在该基于云的系统环境中,可以将各种服务作为云服务来提供。在图22所示的示例中,云基础设施系统2202可以提供可由使用一个或多个客户端计算设备2204、2206和2208的用户请求的一个或多个云服务。云基础设施系统2202可以包括一个或多个计算机和/或服务器,其可以包括以上针对服务器2112描述的那些计算机和/或服务器。云基础设施系统2202中的计算机可以被组织为通用计算机、专用服务器计算机、服务器群、服务器集群或任何其他适当的布置和/或组合。

[0183] 网络2210可以有助于客户端2204、2206和2208与云基础设施系统2202之间的数据通信和交换。网络2210可以包括一个或多个网络。网络可以是相同类型的,也可以是不同类型的。网络2210可以支持一个或多个通信协议,包括有线和/或无线协议,以便于通信。

[0184] 图22中描绘的示例仅是云基础设施系统的一个示例,并不旨在是限制性的。应当理解,在一些其他示例中,云基础设施系统2202可以具有比图22中描绘的组件更多或更少的组件,可以组合两个或更多组件,或者可以具有不同的组件配置或布置。例如,尽管图22描绘了三个客户端计算设备,但是在备选示例中可以支持任意数量的客户端计算设备。

[0185] 术语云服务通常用于指由服务提供商的系统(例如,云基础设施系统2202)按需并经由诸如互联网的通信网络向用户提供的服务。通常,在公共云环境中,组成云服务提供商系统的服务器和系统不同于客户自己的企业服务器和系统。云服务提供商的系统由云服务提供商管理。因此,客户可以利用云服务提供商提供的云服务,而不必为这些服务购买单独的许可证、支持或硬件和软件资源。例如,云服务提供商的系统可以托管应用,并且用户可以经由互联网按需订购和使用应用,而不必购买用于执行应用的基础设施资源。云服务旨在提供对应用、资源和服务的轻松、可扩展访问。若干提供商提供云服务。例如,若干云服务由加利福尼亚州红木海岸的Oracle Corporation® 提供,例如中间件服务、数据库服务、Java云服务等。

[0186] 在某些实施例中,云基础设施系统2202可以使用诸如在软件即服务(SaaS)模型、平台即服务(PaaS)模型、基础设施即服务(IaaS)模型以及包括混合服务模型的其他模型下的不同模型来提供一个或多个云服务。云基础设施系统2202可以包括能够提供各种云服务的一套应用、中间件、数据库和其他资源。

[0187] SaaS模型使应用或软件能够通过通信网络(如互联网)作为服务交付给客户,而客户不必购买针对底层应用的硬件或软件。例如,SaaS模型可用于向客户提供对由云基础设施系统2202托管的按需应用的访问。Oracle Corporation®提供的SaaS服务的示例包括但不限于针对人力资源/资本管理、客户关系管理(CRM)、企业资源规划(ERP)、供应链管理(SCM)、企业绩效管理(EPM)、分析服务、社交应用等的各种服务。

[0188] IaaS模式通常用于将基础设施资源(例如服务器、存储、硬件和网络资源)以云服务的形式提供给客户,以提供弹性计算和存储能力。Oracle Corporation®提供各种IaaS服务。

[0189] PaaS模型通常用于提供作为服务的平台和环境资源,使客户能够开发、运行和管理应用和服务,而不必采购、构建或维护此类资源。Oracle Corporation®提供的PaaS服务的示例包括但不限于Oracle Java Cloud Service(JCS)、Oracle Database Cloud Service(DBCS)、数据管理云服务、各种应用开发解决方案服务等。

[0190] 云服务通常基于按需自助服务、基于订阅、可弹性扩展、可靠、高度可用和安全的方式提供。例如,客户可以经由订阅订单来订购由云基础设施系统2202提供的一个或多个服务。然后,云基础设施系统2202执行处理以提供客户的订阅订单中请求的服务。云基础设施系统2202可以被配置为提供一个或多个云服务。

[0191] 云基础设施系统2202可以经由不同的部署模式提供云服务。在公共云模型中,云基础设施系统2202可以由第三方云服务提供商拥有,并且云服务被提供给任何一般公共客户,其中客户可以是个人或企业。在私有云模型下,云基础设施系统2202可以在组织内(例如,在企业组织内)运行,并向该组织内的客户提供服务。例如,客户可以是企业的各个部门,例如人力资源部、薪资部等,甚至可以是企业内部的个人。在社区云模式下,云基础设施系统2202和所提供的服务可以由相关社区中的几个组织共享。也可以使用各种其他模型,例如上述模型的混合。

[0192] 客户端计算设备2204、2206和2208可以是不同类型的(例如图21中描绘的设备2102、2104、2106和2108),并且能够操作一个或多个客户端应用。用户可以使用客户端设备与云基础设施系统2202交互,诸如请求云基础设施系统2202提供的服务。

[0193] 在一些实施例中,云基础设施系统2202执行的用于提供管理相关服务的处理可以涉及大数据分析。这种分析可以涉及使用、分析和操作大型数据集,以检测和可视化数据中的各种趋势、行为、关系等。该分析可以由一个或多个处理器执行,可能并行处理数据、使用数据执行模拟等。例如,可以由云基础设施系统2202执行大数据分析,用于以自动方式确定正则表达式。用于该分析的数据可以包括结构化数据(例如,存储在数据库中或根据结构化模型结构化的数据)和/或非结构化数据(例如,数据斑点(二进制大对象))。

[0194] 如图22中的示例所示,云基础设施系统2202可以包括基础设施资源2230,其用于协助由云基础设施系统2202提供的各种云服务的提供。基础设施资源2230可以包括例如处

理资源、存储或存储器资源、联网资源等。

[0195] 在某些实施例中,为了便于高效地提供这些资源以支持云基础设施系统2202为不同客户提供的各种云服务,可以将这些资源捆绑成资源集或资源模块(也称为“pod”)。每个资源模块或pod可以包括一种或多种类型的资源的预先集成和优化组合。在某些实施例中,可以为不同类型的云服务预先提供不同的pod。例如,可以为数据库服务供应第一组pod,可以为Java服务提供第二组pod,其可以包括与第一组pod中的pod不同的资源组合等等。对于一些服务,为提供服务而分配的资源可以在服务之间共享。

[0196] 云基础设施系统2202自身可以内部使用由云基础设施系统2202的不同组件共享并且便于云基础设施系统2202提供服务的服务2232。这些内部共享服务可以包括但不限于,安全和身份服务、集成服务、企业存储库服务、企业管理器服务、病毒扫描和白名单服务、高可用性、备份和恢复服务、用于启用云支持的服务、电子邮件服务、通知服务、文件传输服务等等。

[0197] 云基础设施系统2202可以包括多个子系统。这些子系统可以以软件或硬件或其组合来实现。如图22所示,子系统可以包括使云基础设施系统2202的用户或客户能够与云基础设施系统2202交互的用户界面子系统2212。用户界面子系统2212可以包括各种不同的界面,诸如web界面2214、其中由云基础设施系统2202提供的云服务被广告并且可由消费者购买的在线商店界面2216,以及其他界面2218。例如,客户可以使用客户端设备,使用接口2214、2216和2218中的一个或多个请求(服务请求2234)由云基础设施系统2202提供的一个或多个服务。例如,客户可以访问在线商店,浏览云基础设施系统2202提供的云服务,并对客户希望订阅的云基础设施系统2202提供的一个或多个服务下订阅订单。服务请求可以包括标识客户的信息以及客户希望订阅的一个或多个服务。例如,客户可以订阅订购云基础设施系统2202提供的自动生成正则表达式相关的服务。

[0198] 在某些实施例中,例如图22中描绘的示例,云基础设施系统2202可以包括被配置为处理新订单的订单管理子系统(OMS)2220。作为该处理的一部分,OMS 2220可以被配置为:创建针对客户的账户(如果还没有的话);从客户接收账单和/或账户信息,该信息将用于向向客户提供所请求的服务的客户收费;验证客户信息;在验证之后,为客户预订订单;以及协调各种工作流以准备订单以供供应。

[0199] 一旦正确验证,OMS 2220然后可以调用订单供应子系统(OPS)2224,其被配置为供应针对订单的资源,包括处理、存储器和联网资源。供应可以包括为订单分配资源以及配置资源以协助客户订单所请求的服务。针对订单供应资源的方式以及供应的资源的类型可以取决于客户已经订购的云服务的类型。例如,根据一个工作流,OPS2224可以被配置为确定被请求的特定云服务并标识可能已经针对该特定云服务预先配置的多个pod。针对订单分配的pod数量可以取决于所请求服务的大小/金额/级别/范围。例如,可以基于服务要支持的用户数量、请求服务的持续时间等来确定要分配的pod的数量。然后,可以针对特定的请求客户定制所分配的pod,以提供所请求的服务。

[0200] 云基础设施系统2202可以向请求客户发送响应或通知2244,以指示所请求的服务何时准备好以供使用。在一些情况下,可以向客户发送使客户能够开始使用和利用所请求的服务的好处的信息(例如,链接)。在某些实施例中,对于请求自动生成正则表达式相关服务的客户,响应可以包括在执行时导致显示用户界面的指令。

[0201] 云基础设施系统2202可以向多个客户提供服务。对于每个客户,云基础设施系统2202负责管理与从客户接收的一个或多个订阅订单相关的信息,维护与订单相关的客户数据,以及向客户提供所请求的服务。云基础设施系统2202还可以收集关于客户对订阅服务的使用的使用统计。例如,可以收集关于所使用的存储量、传输的数据量、用户数、以及系统运行时间和系统停机时间的量等的统计数据。该使用信息可用于向客户计费。例如,可以按月周期计费。

[0202] 云基础设施系统2202可以并行地向多个客户提供服务。云基础设施系统2202可以为这些客户存储信息,可能包括专有信息。在某些实施例中,云基础设施系统2202包括身份管理子系统(IMS)2228,其被配置为管理客户信息并提供所管理信息的分离,使得与一个客户相关的信息不能被另一个客户访问。IMS2228可以被配置为提供各种安全相关服务,例如身份服务;信息访问管理、认证和授权服务;用于管理客户身份和角色以及相关能力的服务等。

[0203] 图23示出了计算机系统2300的示例。在一些实施例中,计算机系统2300可以用于实现上述任何系统。如图23所示,计算机系统2300包括各种子系统,包括经由总线子系统2302与多个其他子系统通信的处理子系统2304。这些其他子系统可以包括处理加速单元2306、I/O子系统2308、存储子系统2318和通信子系统2324。存储子系统2318可以包括非暂态计算机可读存储介质,包括存储介质2322和系统存储器2310。

[0204] 总线子系统2302提供用于让计算机系统2300的各种组件和子系统按预期彼此通信的机制。尽管总线子系统2302示意性地示出为单条总线,但是总线子系统的备选示例可以利用多条总线。总线子系统2302可以是几种类型的总线结构中的任何一种,包括存储器总线或存储器控制器、外围设备总线、使用各种总线体系结构中的任何一种的局部总线等。例如,这样的体系结构可以包括工业标准体系结构(ISA)总线、微信道体系结构(MCA)总线、增强型ISA(EISA)总线、视频电子标准协会(VESA)局部总线和外围组件互连(PCI)总线,其可以被实现为按照IEEE P1386.1标准制造的夹层总线等。

[0205] 处理子系统2304控制计算机系统2300的操作,并且可以包括一个或多个处理器、专用集成电路(ASIC)或现场可编程门阵列(FPGA)。处理器可以包括单核或多核处理器。计算机系统2300的处理资源可以被组织成一个或多个处理单元2332、2334等。处理单元可以包括一个或多个处理器、来自相同或不同处理器的一个或多个核、核和处理器的组合、或核和处理器的其他组合。在一些实施例中,处理子系统2304可以包括一个或多个专用协处理器,例如图形处理器、数字信号处理器(DSP)等。在一些实施例中,处理子系统2304的一些或全部处理单元可以使用定制电路(例如专用集成电路(ASIC)或现场可编程门阵列(FPGA))来实现。

[0206] 在一些实施例中,处理子系统2304中的处理单元可以执行存储在系统存储器2310或计算机可读存储介质2322上的指令。在各种示例中,处理单元可以执行各种程序或代码指令,并且可以维护多个并发执行的程序或进程。在任何给定时间,要执行的程序代码中的一些或全部可以驻留在系统存储器2310和/或计算机可读存储介质2322上,包括可能在一个或多个存储设备上。通过适当的编程,处理子系统2304可以提供上述各种功能。在计算机系统2300正在执行一个或多个虚拟机的情况下,可以将一个或多个处理单元分配给每个虚拟机。

[0207] 在某些实施例中,处理加速单元2306可以可选地用于执行定制处理或用于卸载由处理子系统2304执行的一些处理,从而加速由计算机系统2300执行的整体处理。

[0208] I/O子系统2308可以包括用于向计算机系统2300输入信息和/或用于从或经由计算机系统2300输出信息的设备和机制。通常,术语输入设备的使用意在包括用于向计算机系统2300输入信息的所有可能类型的设备和机制。用户界面输入设备可以包括例如键盘、诸如鼠标或轨迹球之类的定点设备、集成到显示器中的触摸板或触摸屏、滚轮、点击轮、拨盘、按钮、开关、小键盘、具有语音命令识别系统的音频输入设备、麦克风以及其他类型的输入设备。用户界面输入设备还可以包括动作感测和/或手势识别设备,例如使用户能够控制输入设备并与其交互的Microsoft **Kinect®**运动传感器、Microsoft **Xbox®**360游戏控制器、提供使用手势和口头命令接收输入的接口的设备。用户界面输入设备还可以包括眼势识别设备,例如Google **Glass®**眨眼检测器,其检测来自用户的眼部活动(例如,在拍照和/或进行菜单选择时的"眨眼"),并将眼势转换为输入设备(例如,Google **Glass®**)的输入。此外,用户界面输入设备可以包括使用户能够通过语音命令与语音识别系统(例如,**Siri®**导航器)交互的语音识别感测设备。

[0209] 用户界面输入设备的其他示例包括但不限于,三维(3D)鼠标、操纵杆或指点杆、游戏板和图形输入板,以及诸如扬声器、数码相机、数码摄像机、便携式媒体播放器、网络摄像机、图像扫描仪、指纹扫描仪、条形码读取器3D扫描仪、3D打印机、激光测距仪和眼睛注视追踪设备的音频/视觉设备。此外,用户界面输入设备可以包括例如医学成像输入设备,诸如计算机断层扫描、磁共振成像、位置发射断层扫描和医学超声成像设备。用户界面输入设备还可以包括例如音频输入设备,诸如MIDI键盘、数字乐器等。

[0210] 通常,术语输出设备的使用意在包括用于将信息从计算机系统2300输出到用户或其他计算机的所有可能类型的设备和机制。用户界面输出设备可以包括显示子系统、指示灯或诸如音频输出设备等的非可视显示器等。显示子系统可以是阴极射线管(CRT)、平板设备(例如,使用液晶显示器(LCD)或等离子显示器的平板设备)、投影设备、触摸屏等。例如,用户界面输出设备可以包括但不限于各种可视地传达文本、图形和音频/视频信息的显示设备,诸如监视器、打印机、扬声器、耳机、汽车导航系统、绘图仪、语音输出设备和调制解调器。

[0211] 存储子系统2318提供用于存储由计算机系统2300使用的信息和数据的存储库或数据存储。存储子系统2318提供有形的非暂态计算机可读存储介质,用于存储提供某些示例的功能的基本编程和数据构造。存储子系统2318可以存储当由处理子系统2304执行时提供上述功能的软件(例如,程序、代码模块、指令)。该软件可以由处理子系统2304的一个或多个处理单元执行。存储子系统2318还可以提供用于存储根据本公开的教导使用的数据的存储库。

[0212] 存储子系统2318可以包括一个或多个非易失性存储设备,包括易失性和非易失性存储设备。如图23所示,存储子系统2318包括系统存储器2310和计算机可读存储介质2322。系统存储器2310可以包括多个存储器,包括用于在程序执行期间存储指令和数据的易失性主随机存取存储器(RAM)和其中存储固定指令的非易失性只读存储器(ROM)或闪存。在一些实现中,基本输入/输出系统(BIOS)通常可以存储在ROM中,该基本输入/输出系统(BIOS)包

含诸如在启动期间帮助在计算机系统2300内的元件之间传输信息的基本例程。RAM通常包含当前由处理子系统2304操作和执行的数据和/或程序模块。在一些实现中，系统存储器2310可以包括多种不同类型的存储器，诸如静态随机存取存储器(SRAM)、动态随机存取存储器(DRAM)等。

[0213] 作为示例而非限制，如图23所示，系统存储器2310可以加载正在执行的应用程序2312，该应用程序2312可以包括诸如web浏览器、中间层应用、关系数据库管理系统(RDBMS)等的各种应用，程序数据2314和操作系统2316。作为示例，操作系统2316可以包括各种版本的Microsoft Windows®、Apple Macintosh®和/或Linux操作系统、各种市售的UNIX®或类UNIX操作系统(包括但不限于各种GNU/Linux操作系统、Google Chrome® OS等)和/或诸如iOS、Windows® Phone、Android® OS、BlackBerry® OS、Palm® OS操作系统等的移动操作系统。

[0214] 计算机可读存储介质2322可以存储提供一些示例的功能的编程和数据构造。计算机可读介质2322可以为计算机系统2300提供计算机可读指令、数据结构、程序模块和其他数据的存储。当由处理子系统2304执行时提供上述功能的软件(程序、代码模块、指令)可以存储在存储子系统2318中。举例来说，计算机可读存储介质2322可以包括非易失性存储器，例如硬盘驱动器、磁盘驱动器、光盘驱动器(例如，CD ROM、DVD、Blu-Ray®盘)或其他光学介质。计算机可读存储介质2322可以包括但不限于，Zip®驱动器、闪存卡、通用串行总线(USB)闪存驱动器、安全数字(SD)卡、DVD盘、数字录像带等。计算机可读存储介质2322还可以包括基于非易失性存储器的固态驱动器(SSD)(诸如基于闪存的SSD、企业闪存驱动器、固态ROM等)、基于易失性存储器的SSD(诸如固态RAM、动态RAM、静态RAM、基于DRAM的SSD、磁阻RAM(MRAM)SSD)、以及使用DRAM和基于闪存的SSD的组合的混合SSD。

[0215] 在某些实施例中，存储子系统2318还可以包括计算机可读存储介质读取器2320，其还可以连接到计算机可读存储介质2322。读取器2320可以接收来自诸如盘、闪存驱动器等的存储设备的数据并被配置为从该存储设备读取数据。

[0216] 在某些实施例中，计算机系统2300可以支持虚拟化技术，包括但不限于处理和存储器资源的虚拟化。例如，计算机系统2300可以提供对执行一个或多个虚拟机的支持。在某些实施例中，计算机系统2300可以执行便于配置和管理虚拟机的诸如管理程序之类的程序。每个虚拟机可以被分配存储器、计算(例如，处理器、核)、I/O和联网资源。每个虚拟机通常独立于其他虚拟机运行。虚拟机通常运行其自己的操作系统，该操作系统可以与由计算机系统2300执行的其他虚拟机执行的操作系统相同或不同。因此，计算机系统2300可以同时运行多个操作系统。

[0217] 通信子系统2324提供到其他计算机系统和网络的接口。通信子系统2324用作从计算机系统2300接收数据和向其他系统发送数据的接口。例如，通信子系统2324可以使计算机系统2300能够经由互联网建立到一个或多个客户端设备的通信信道，用于从客户端设备接收信息和向客户端设备发送信息。

[0218] 通信子系统2324可以支持有线和/或无线通信协议。在某些实施例中，通信子系统2324可以包括用于接入无线语音和/或数据网络的射频(RF)收发机组件(例如，使用蜂窝电

话技术、高级数据网络技术，诸如3G、4G或EDGE(用于全球演进的增强型数据速率))、WiFi(IEEE 802.XX系列标准、或其他移动通信技术、或其任意组合)、全球定位系统(GPS)接收器组件和/或其他组件。在一些实施例中，除了无线接口之外或代替无线接口，通信子系统2324可以提供有线网络连接(例如，以太网)。

[0219] 通信子系统2324可以以各种形式接收和发送数据。在一些实施例中，除了其他形式之外，通信子系统2324可以接收结构化和/或非结构化数据馈送2326、事件流2328、事件更新2330等形式的输入通信。例如，通信子系统2324可以被配置为从社交媒体网络和/或其他通信服务的用户实时接收(或发送)数据馈送2326，诸如Twitter®馈送、Facebook®更新、诸如丰富站点摘要(RSS)馈送之类的web馈送、和/或来自一个或多个第三方信息源的实时更新。

[0220] 在某些实施例中，通信子系统2324可以被配置为接收连续数据流形式的数据，该连续数据流可以包括实时事件的事件流2328和/或事件更新2330，其本质上可以是连续的或无边界的，没有显式结束。生成连续数据的应用的示例可以包括例如传感器数据应用、金融自动收报机、网络性能测量工具(例如，网络监控和流量管理应用)、点击流分析工具、汽车流量监控等。

[0221] 通信子系统2324还可以被配置为将数据从计算机系统2300传送到其他计算机系统或网络。数据可以以各种不同的形式(例如，结构化和/或非结构化数据馈送2326、事件流2328、事件更新2330等)传送到可以与耦合到计算机系统2300的一个或多个流数据源计算机通信的一个或多个数据库。

[0222] 计算机系统2300可以是各种类型中的一种，包括手持便携式设备(例如，iPhone®蜂窝电话、iPad®计算平板电脑、PDA)、可穿戴设备(例如，Google Glass®头戴式显示器)、个人计算机、工作站、大型机、信息亭、服务器机架或任何其他数据处理系统。由于计算机和网络的不断变化的性质，图23中描绘的对计算机系统2300的描述仅用于具体示例。具有比图23所示的系统更多或更少的组件的许多其他配置也是可能的。基于本文提供的公开和教导，本领域普通技术人员将理解实现各种示例的其他方式和/或方法。

[0223] 虽然已经描述了具体的示例，但是各种修改、改变、替代结构和等同都是可能的。示例不限于在某些特定数据处理环境中操作，而是可以在多个数据处理环境中自由操作。另外，尽管已经使用一系列特定的交易和步骤描述了某些实例，但是对于本领域的技术人员来说，显然这并不是为了限制。尽管一些流程图将操作描述为顺序过程，但许多操作可以并行或并发执行。此外，可以重新安排操作的顺序。流程可以有图中未包括的其他步骤。上述示例的各种特征和方面可以单独或联合使用。

[0224] 此外，虽然已经使用硬件和软件的特定组合描述了某些示例，但是应该认识到，硬件和软件的其他组合也是可能的。某些示例可以仅在硬件中实现，或者仅在软件中实现，或者使用其组合来实现。本文描述的各种处理可以任意组合在同一处理器或不同处理器上实现。

[0225] 在设备、系统、组件或模块被描述为被配置为执行某些操作或功能的情况下，这种配置可以例如通过设计电子电路来执行操作，通过编程可编程电子电路(诸如微处理器)来执行操作，例如通过执行计算机指令或代码，或者被编程为执行存储在非易失性存储器介

质上的代码或指令的处理器或核心,或其任何组合来实现。进程可以使用各种技术进行通信,包括但不限于用于进程间通信的常规技术,并且不同的进程对可以使用不同的技术,或者同一对进程可以在不同的时间使用不同的技术。

[0226] 在本公开中给出了具体细节,以提供对示例的透彻理解。但是,可以在没有这些具体细节的情况下练习示例。例如,公知的电路、工艺、算法、结构和技术在没有不必要的细节的情况下被示出,以避免混淆示例。本说明书仅提供示例,并不打算限制其他示例的范围、适用性或配置。相反,前面对示例的描述将为本领域技术人员提供用于实现各种实例的使能描述。元件的功能和布置可以发生各种变化。

[0227] 因此,说明书和附图应视为说明性的而非限制性的。然而,显而易见的是,可以在不脱离权利要求中陈述的更广泛的精神和范围的情况下对其进行添加、删减、删除以及其他修改和改变。因此,虽然已经描述了具体的示例,但这些示例并不是为了限制。各种修改和等同在所附权利要求的范围内。

[0228] 在前述说明书中,参考其具体实例描述了本公开的各方面,但是本领域技术人员将认识到,本公开并不局限于此。上述公开的各种特征和方面可以单独或联合使用。此外,在不脱离本说明书更广泛的精神和范围的情况下,除了本文描述的那些环境和应用之外,还可以在任何数量的环境和应用中使用实例。因此,说明书和附图应视为说明性的,而不是限制性的。

[0229] 在前述描述中,出于说明的目的,以特定顺序描述方法。应当理解,在备选示例中,可以按照与所描述的顺序不同的顺序来执行这些方法。还应当理解,上述方法可以由硬件组件执行,或者可以在机器可执行指令序列中实现,机器可执行指令序列可以用于使机器,例如通用或专用处理器或逻辑电路,用指令编程来执行这些方法。这些机器可执行指令可以存储在一个或多个机器可读介质上,例如CD-ROM或其他类型的光盘、软盘、ROM、RAM、EPROM、EEPROM、磁卡或光卡、闪存或适合于存储电子指令的其他类型的机器可读介质。备选地,这些方法可以通过硬件和软件的组合来执行。

[0230] 在组件被描述为被配置为执行某些操作的情况下,这种配置可以例如通过设计电子电路或其他硬件来执行操作、通过编程可编程电子电路(例如,微处理器或其他合适的电子电路)来执行操作、或其任意组合来完成。

[0231] 虽然本文详细描述了本申请的说明性实例,但是应当理解,除了现有技术的限制之外,本发明的概念可以以其他方式不同地体现和使用,并且所附权利要求旨在被解释为包括这样的变化。

[0232] 在组件被描述为“被配置为”执行某些操作的情况下,这种配置可以例如通过设计电子电路或其他硬件来执行操作、通过编程可编程电子电路(例如,微处理器或其他合适的电子电路)来执行操作、或其任意组合来完成。

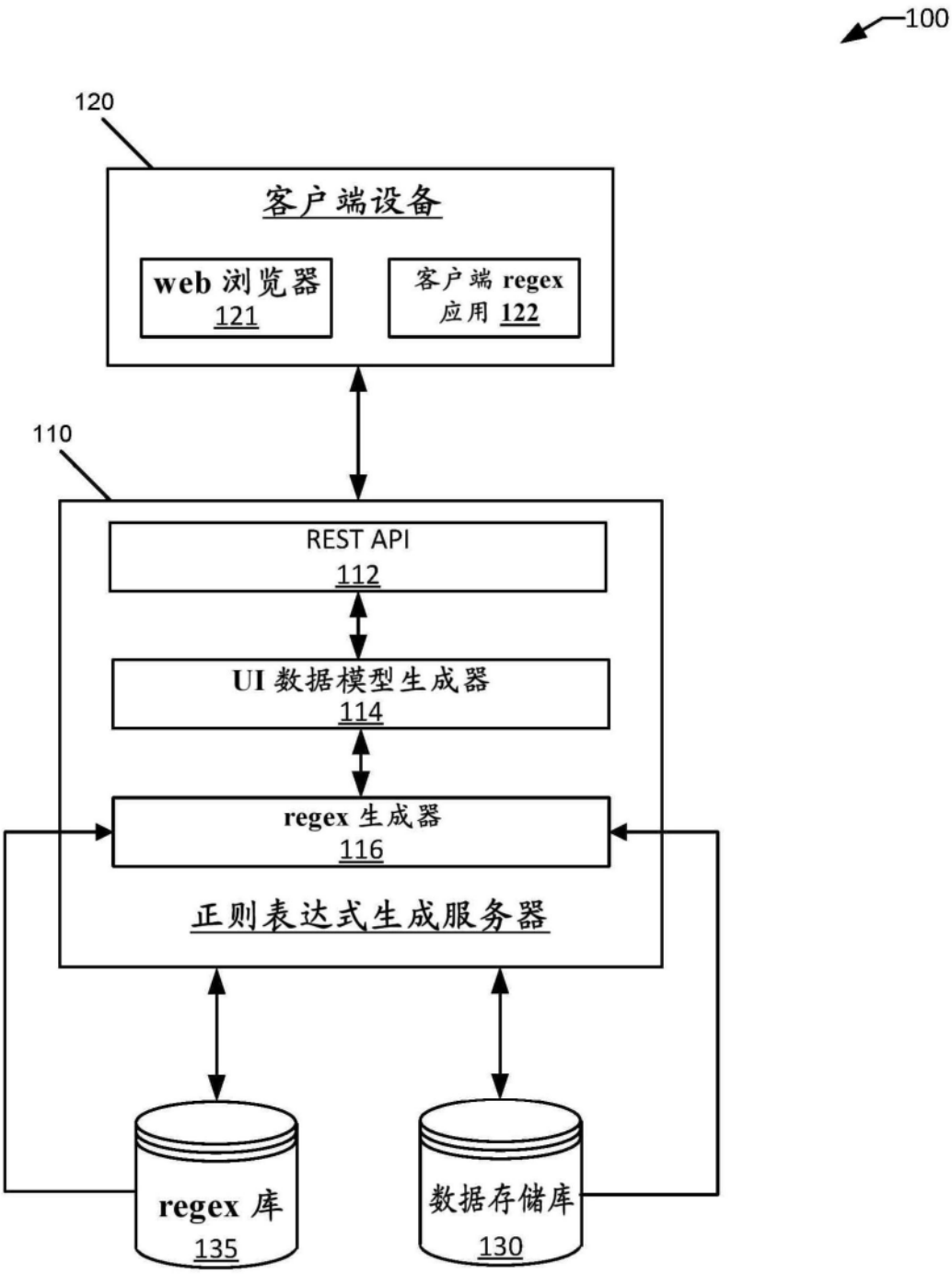

100
120
客户端设备
web 浏览器
121
客户端 regex
应用 122
110
REST API
112
UI 数据模型生成器
114
regex 生成器
116
正则表达式生成服务器
regex 库
135
数据存储库
130

图1

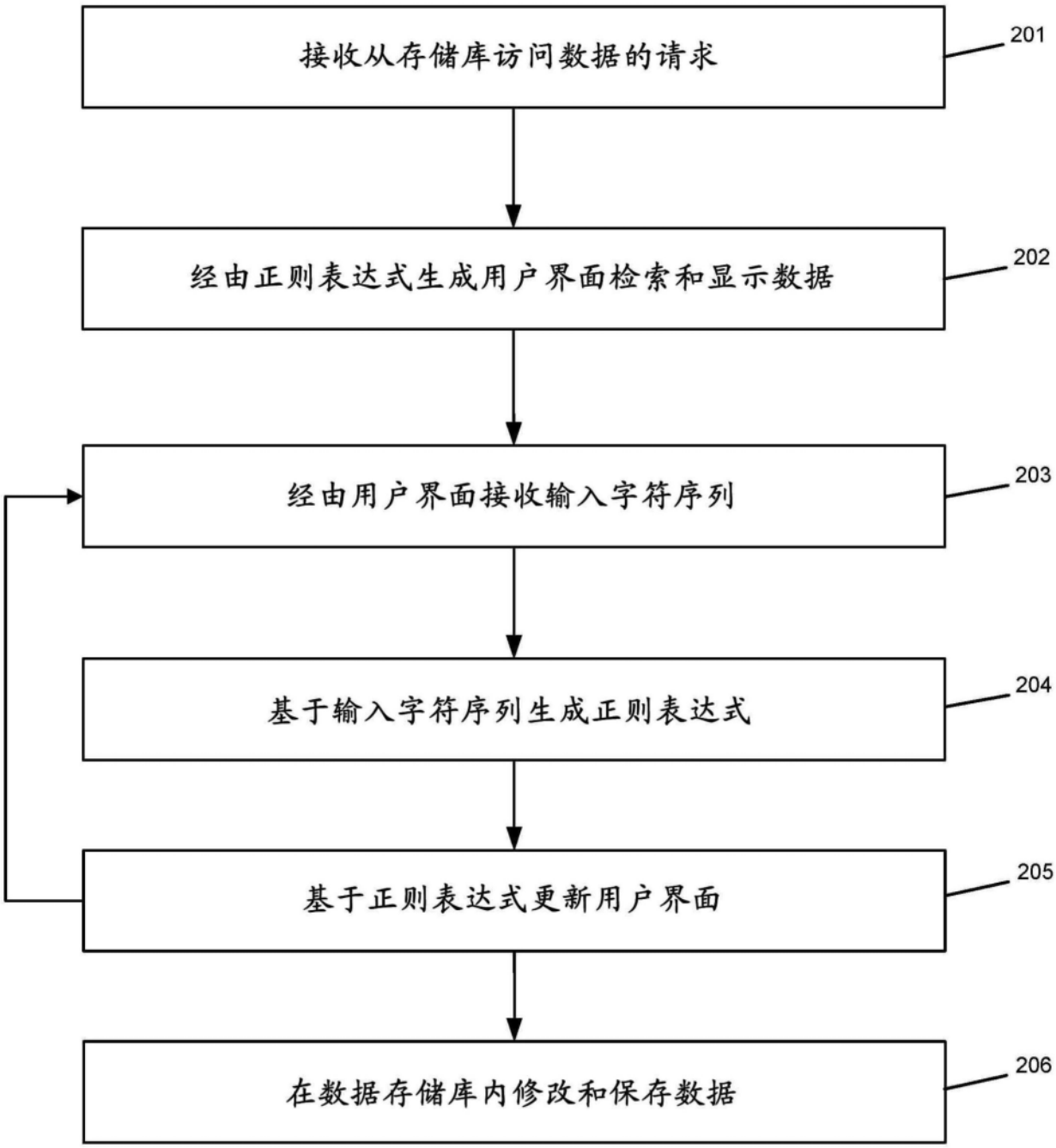

接收从存储库访问数据的请求
201
经由正则表达式生成用户界面检索和显示数据
202
经由用户界面接收输入字符序列
203
基于输入字符序列生成正则表达式
204
基于正则表达式更新用户界面
205
在数据存储库内修改和保存数据
206

图2

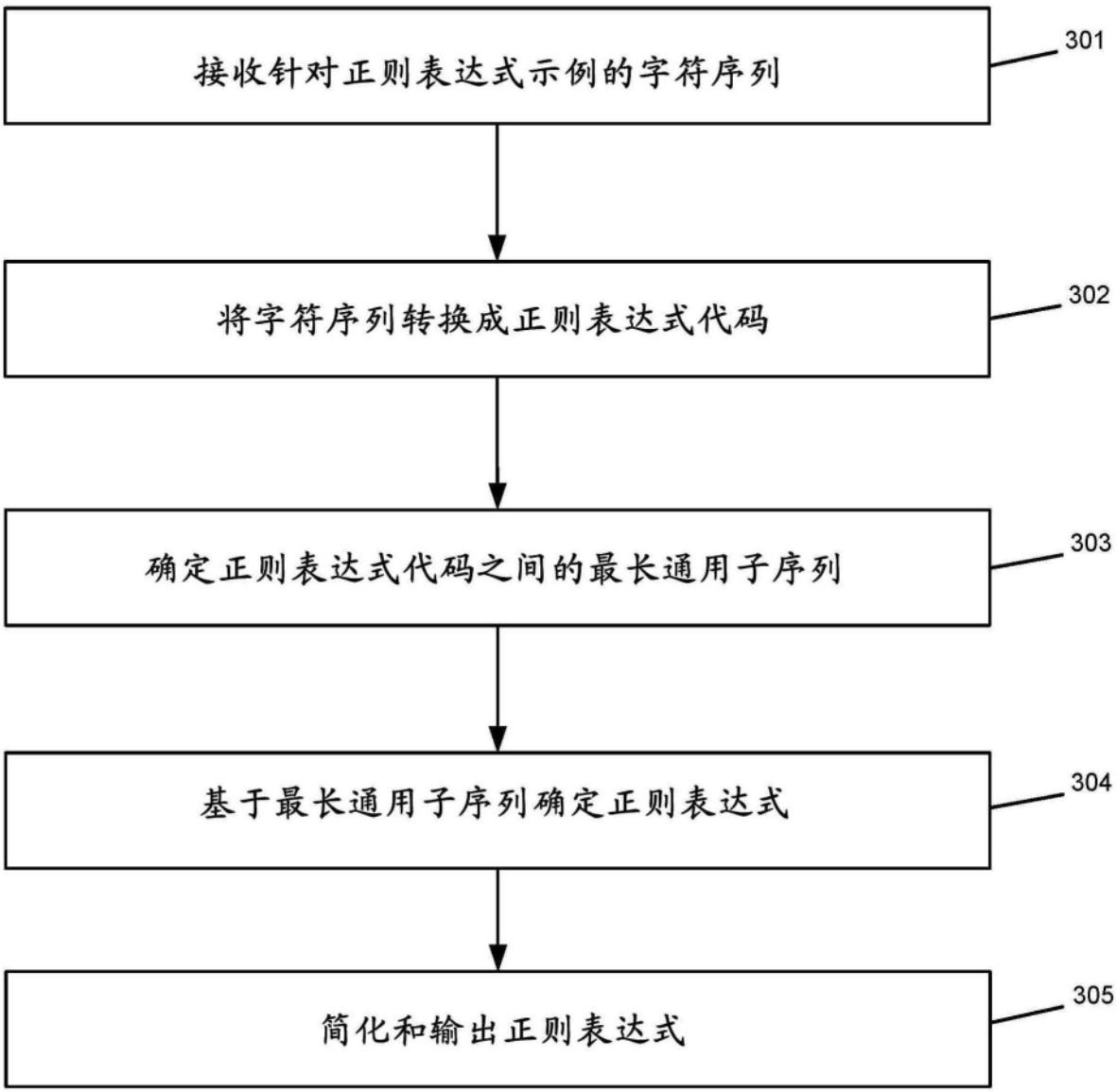

接收针对正则表达式示例的字符序列
301
将字符序列转换成正则表达式代码
302
确定正则表达式代码之间的最长通用子序列
303
基于最长通用子序列确定正则表达式
304
简化和输出正则表达式
305

图3

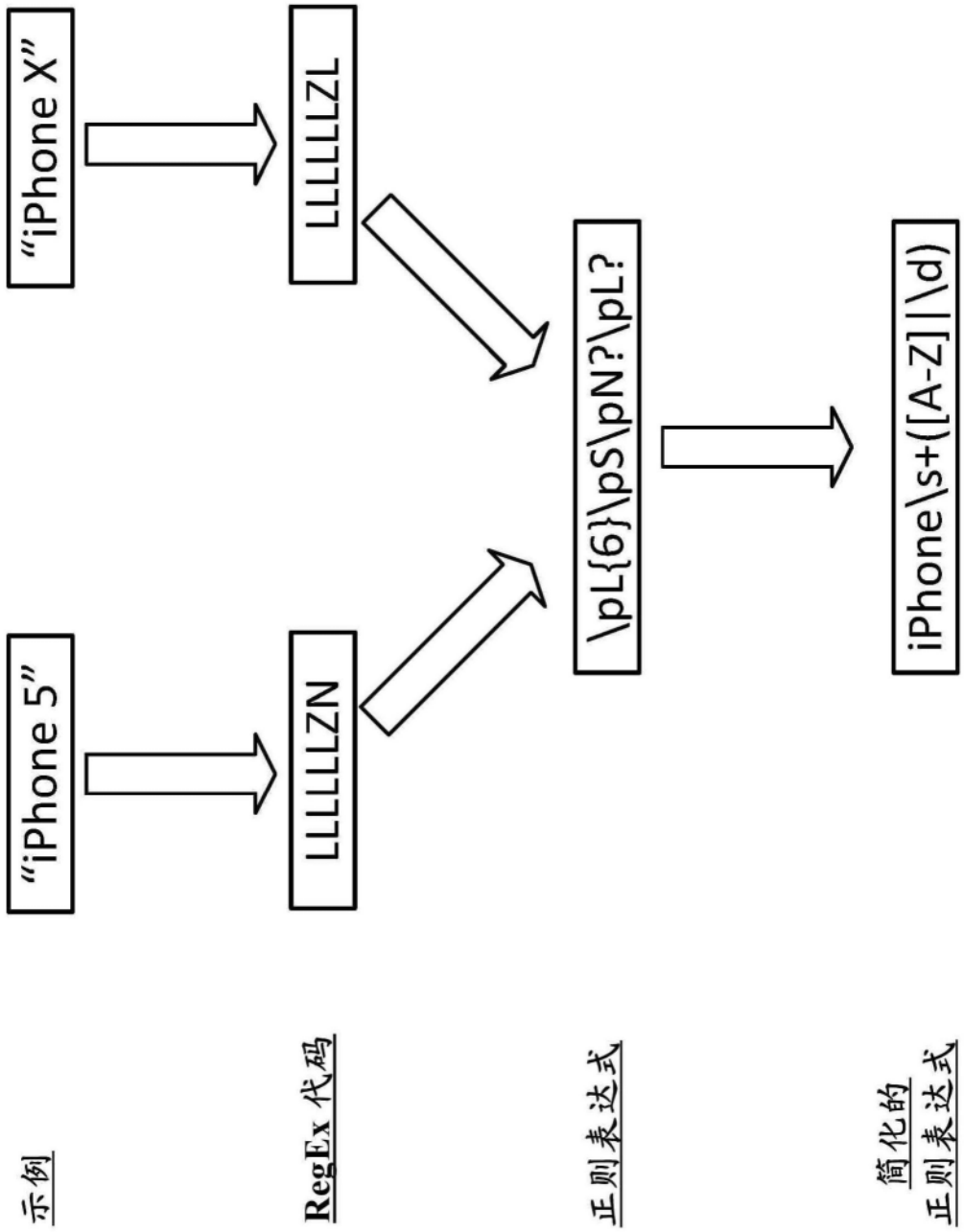

“iPhone X”
“iPhone 5”
LLLLLLZL
LLLLLLZN
\pL{6}\pS\pN?\pL?
iPhone\s+([A-Z]|\d)
示例
RegEx 代码
正则表达式
简化的
正则表达式

图4

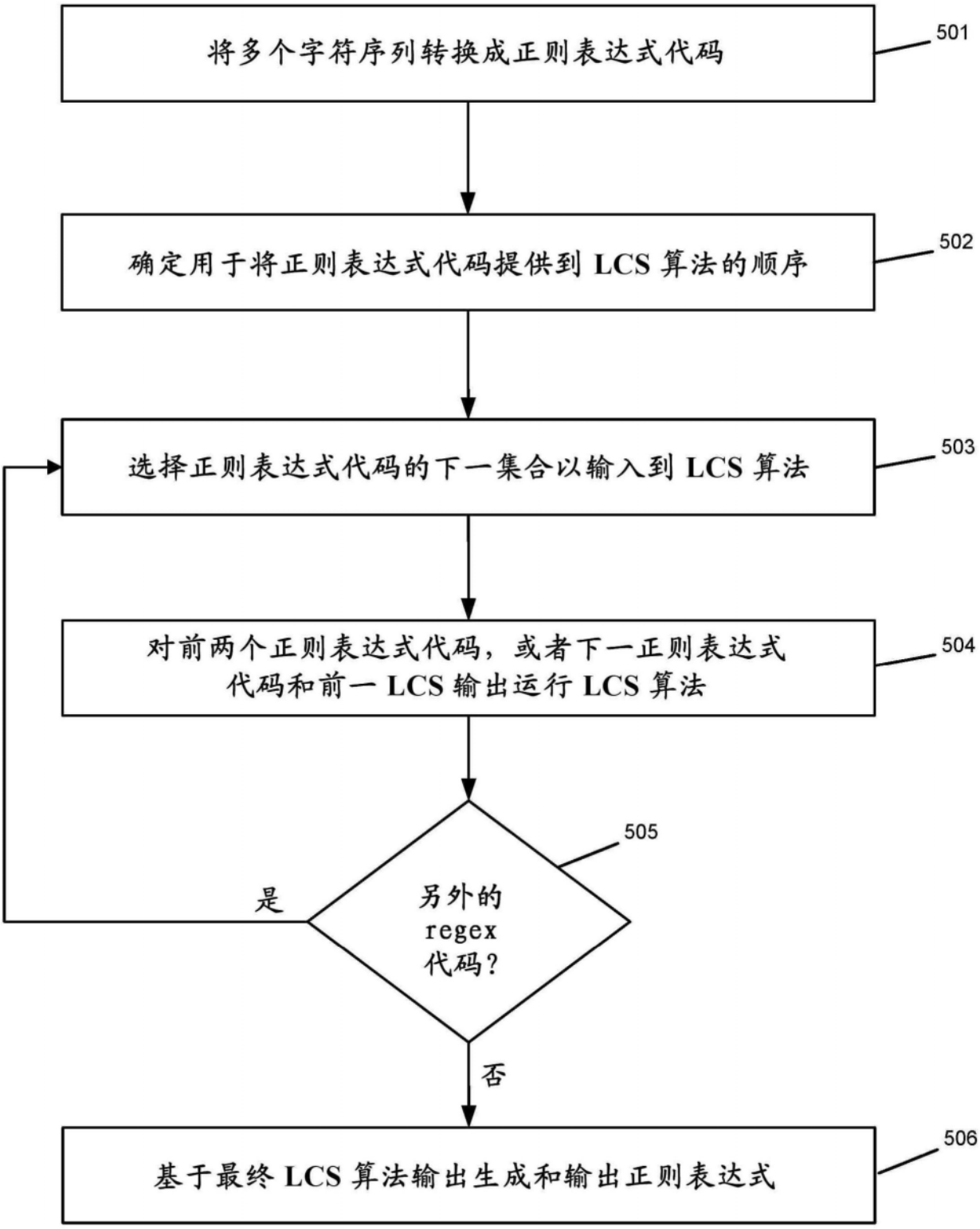

将多个字符序列转换成正则表达式代码
501
确定用于将正则表达式代码提供到 LCS 算法的顺序
502
选择正则表达式代码的下一集合以输入到 LCS 算法
503
对前两个正则表达式代码，或者下一正则表达式代码和前一 LCS 输出运行 LCS 算法
504
505
另外的 regex 代码?
是
否
基于最终 LCS 算法输出生成和输出正则表达式
506

图5

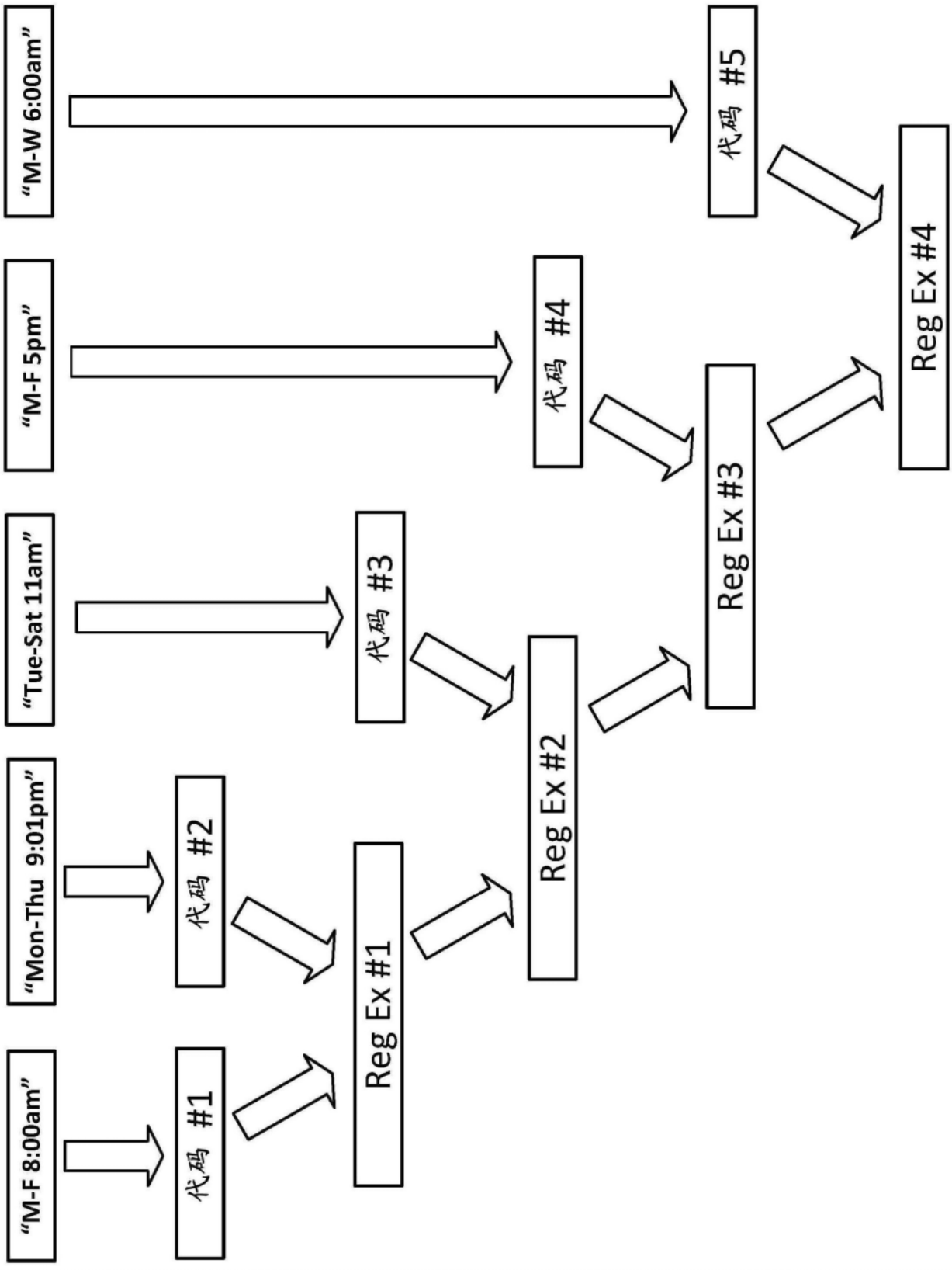

"M-F 8:00am"
"Mon-Thu 9:01pm"
"Tue-Sat 11am"
"M-F 5pm"
"M-W 6:00am"
代码 #1
代码 #2
代码 #3
代码 #4
代码 #5
Reg Ex #1
Reg Ex #2
Reg Ex #3
Reg Ex #4

图6

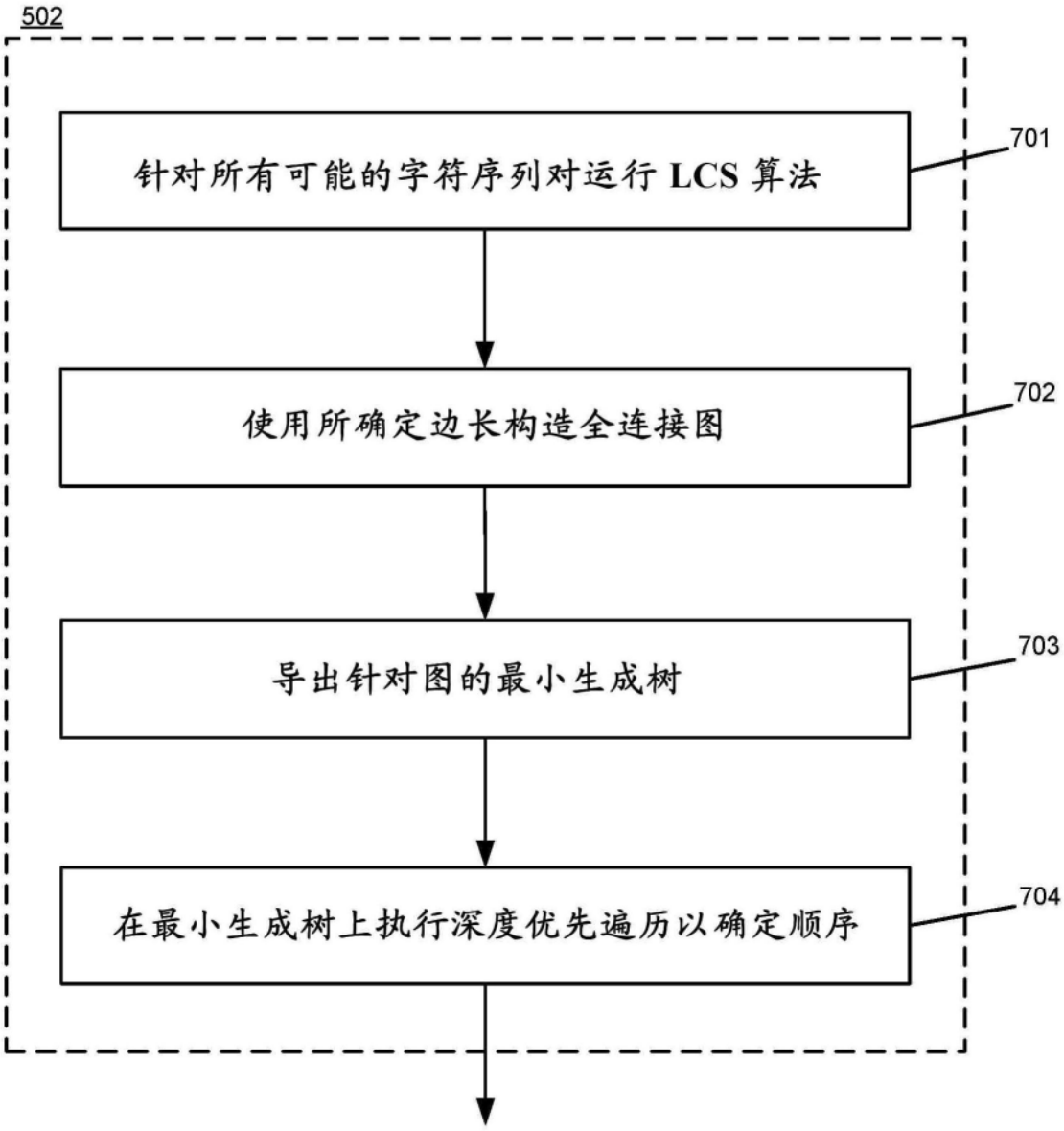

502
针对所有可能的字符序列对运行 LCS 算法
701
使用所确定边长构造全连接图
702
导出针对图的最小生成树
703
在最小生成树上执行深度优先遍历以确定顺序
704

图7

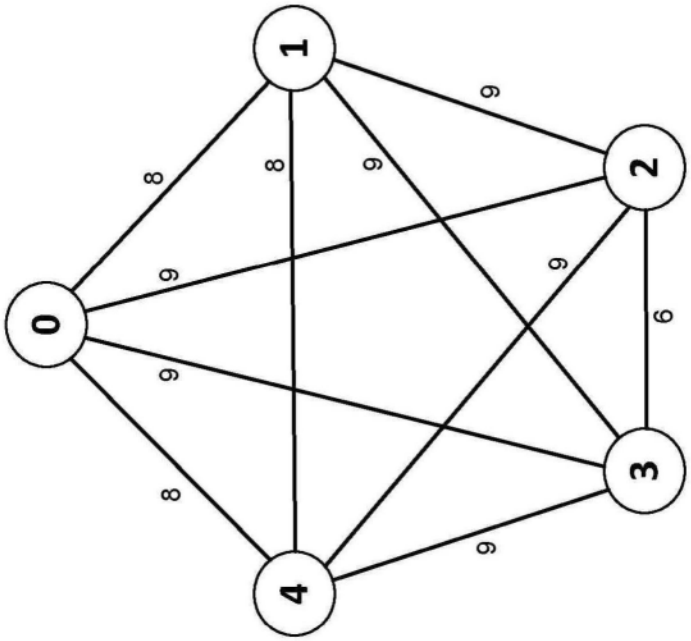


图8A

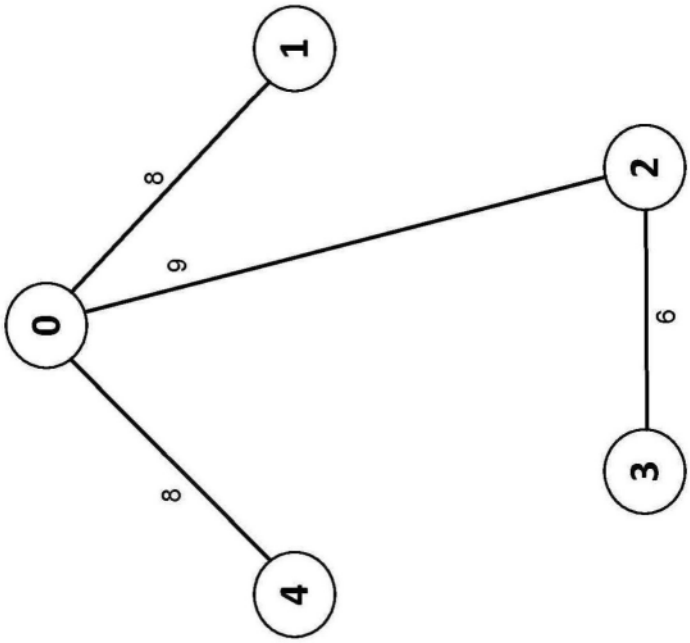


图8B

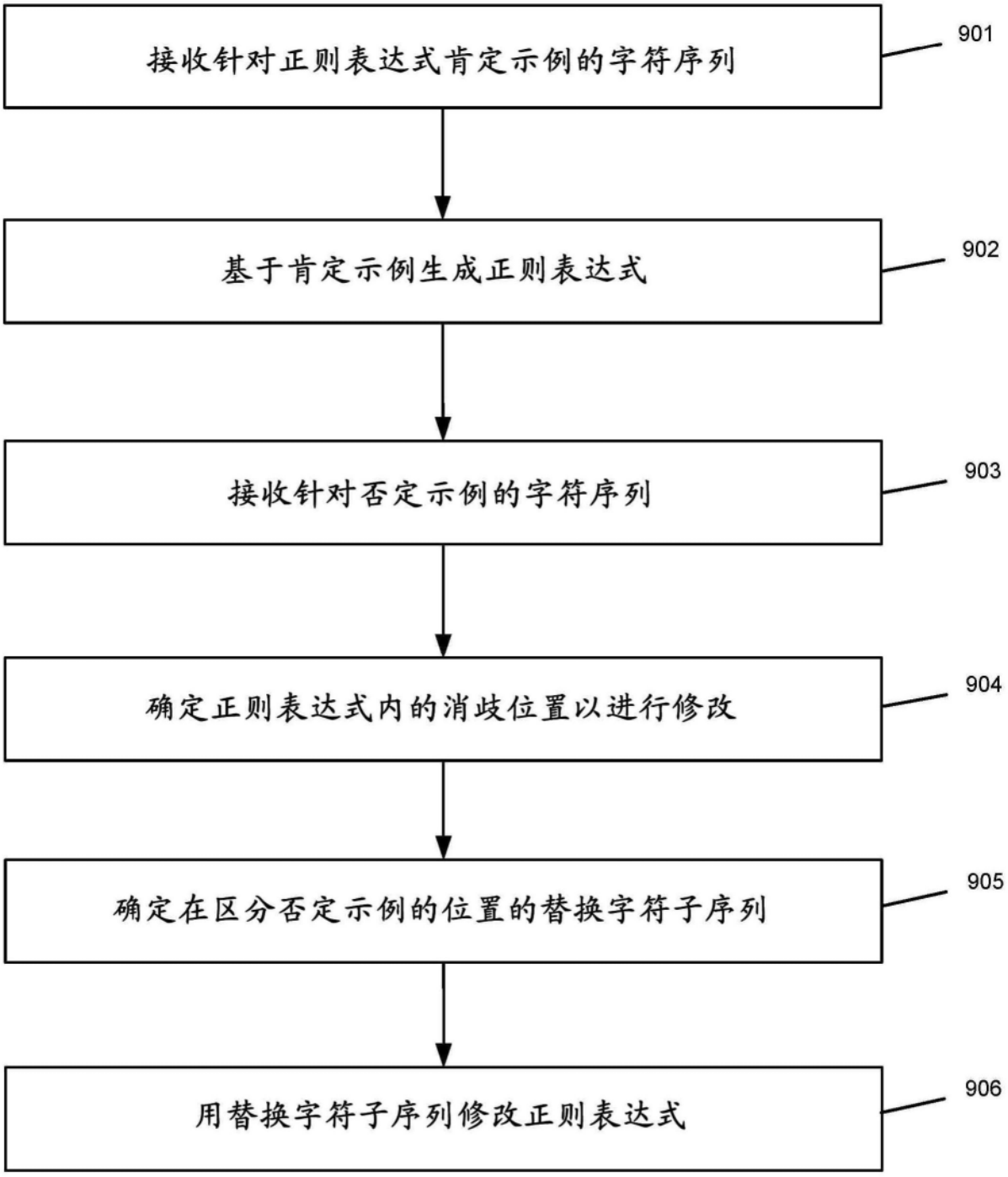

接收针对正则表达式肯定示例的字符序列
901
基于肯定示例生成正则表达式
902
接收针对否定示例的字符序列
903
确定正则表达式内的消歧位置以进行修改
904
确定在区分否定示例的位置的替换字符子序列
905
用替换字符子序列修改正则表达式
906

图9

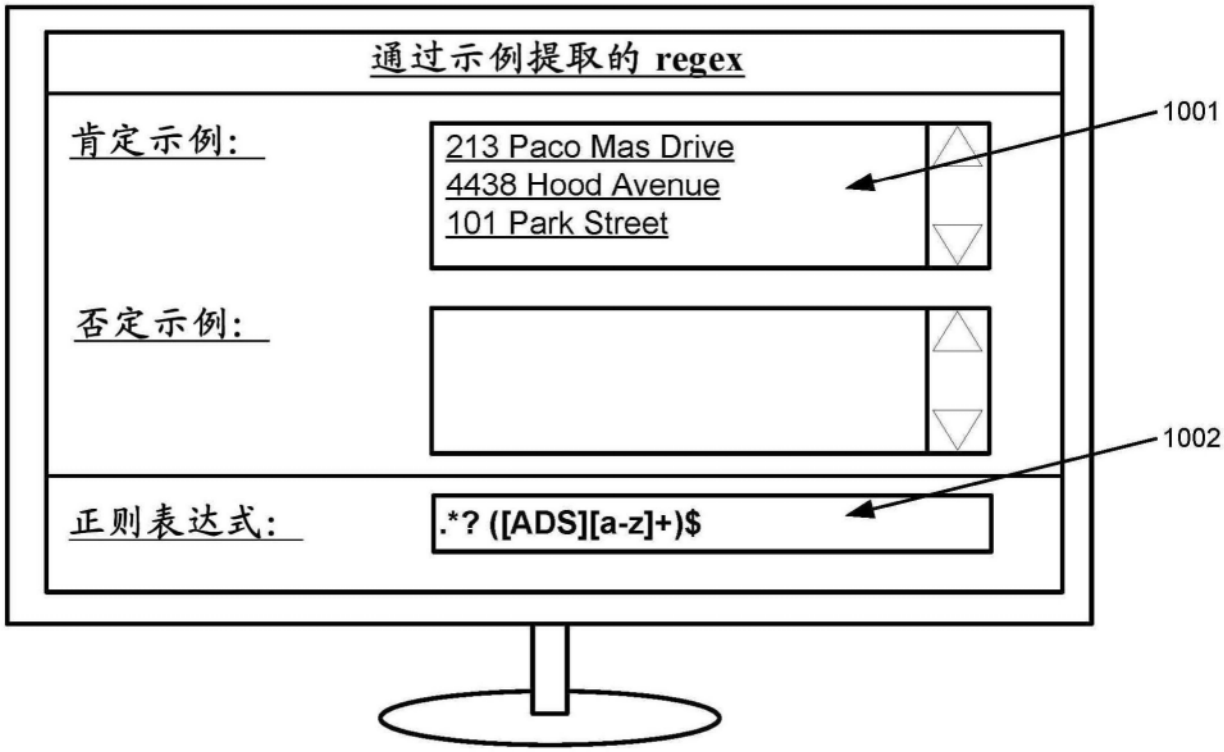

通过示例提取的 regex
肯定示例:
213 Paco Mas Drive
4438 Hood Avenue
101 Park Street
否定示例:
正则表达式:
.*? ([ADS][a-z]+)$
1001
1002

图10A

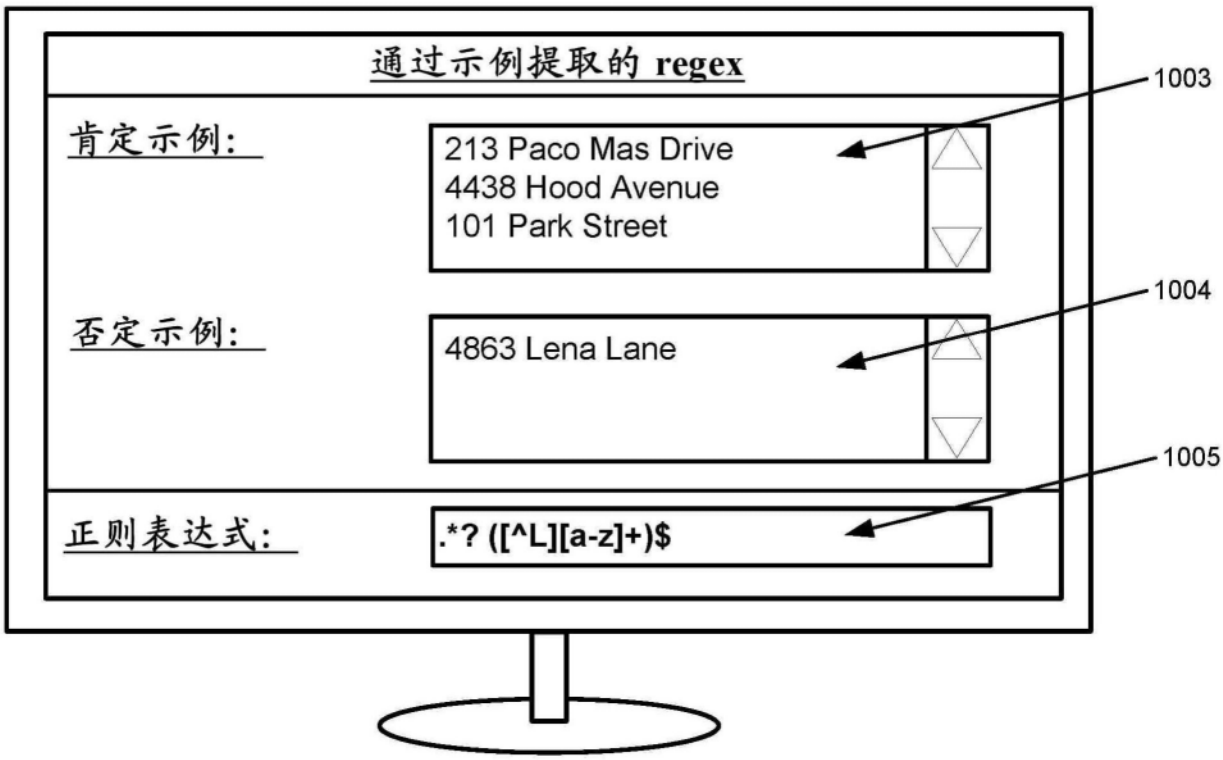

通过示例提取的 regex
肯定示例:
213 Paco Mas Drive
4438 Hood Avenue
101 Park Street
否定示例:
4863 Lena Lane
正则表达式:
.*? ([^L][a-z]+)$
1003
1004
1005

图10B

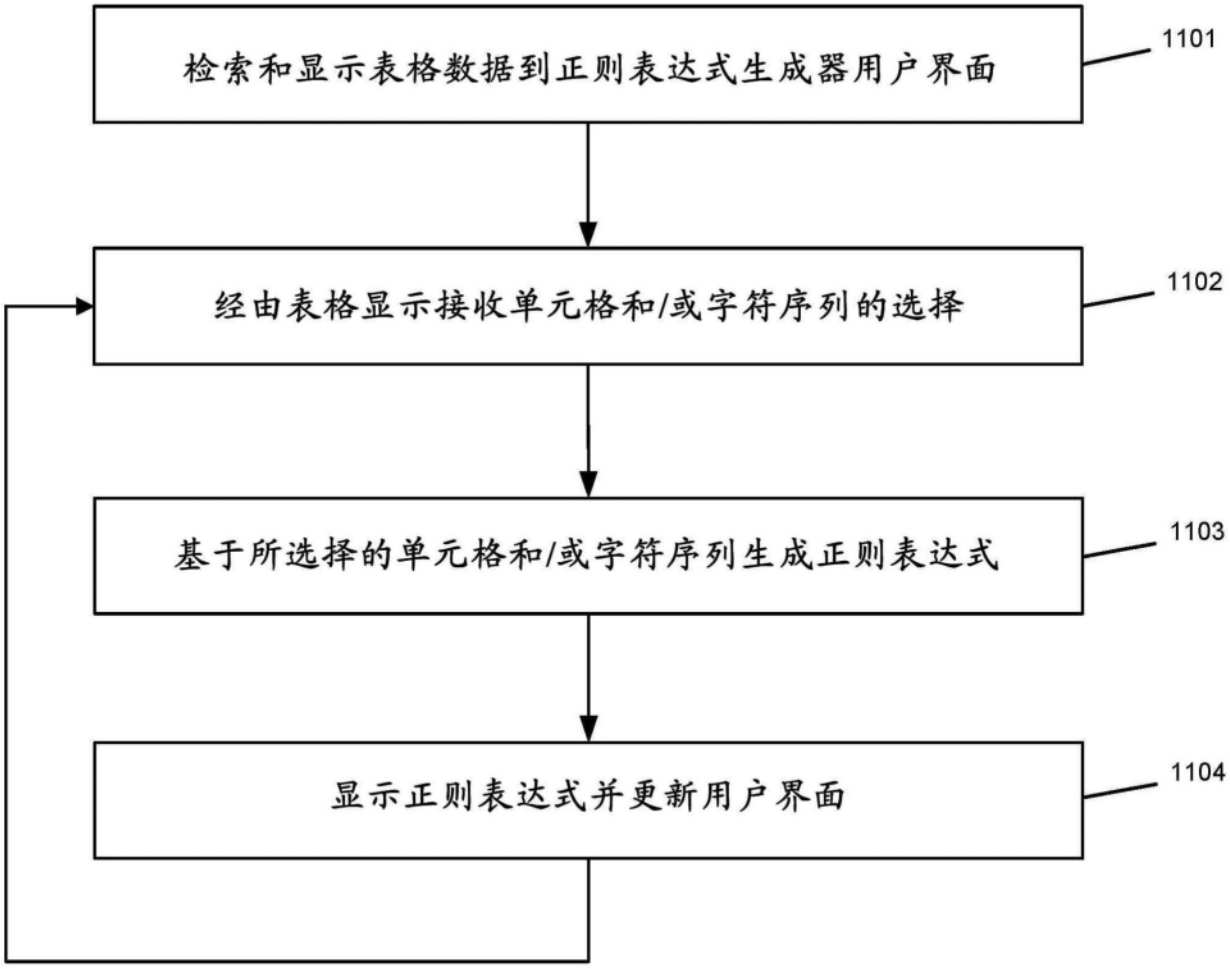

检索和显示表格数据到正则表达式生成器用户界面
1101
经由表格显示接收单元格和/或字符序列的选择
1102
基于所选择的单元格和/或字符序列生成正则表达式
1103
显示正则表达式并更新用户界面
1104

图11

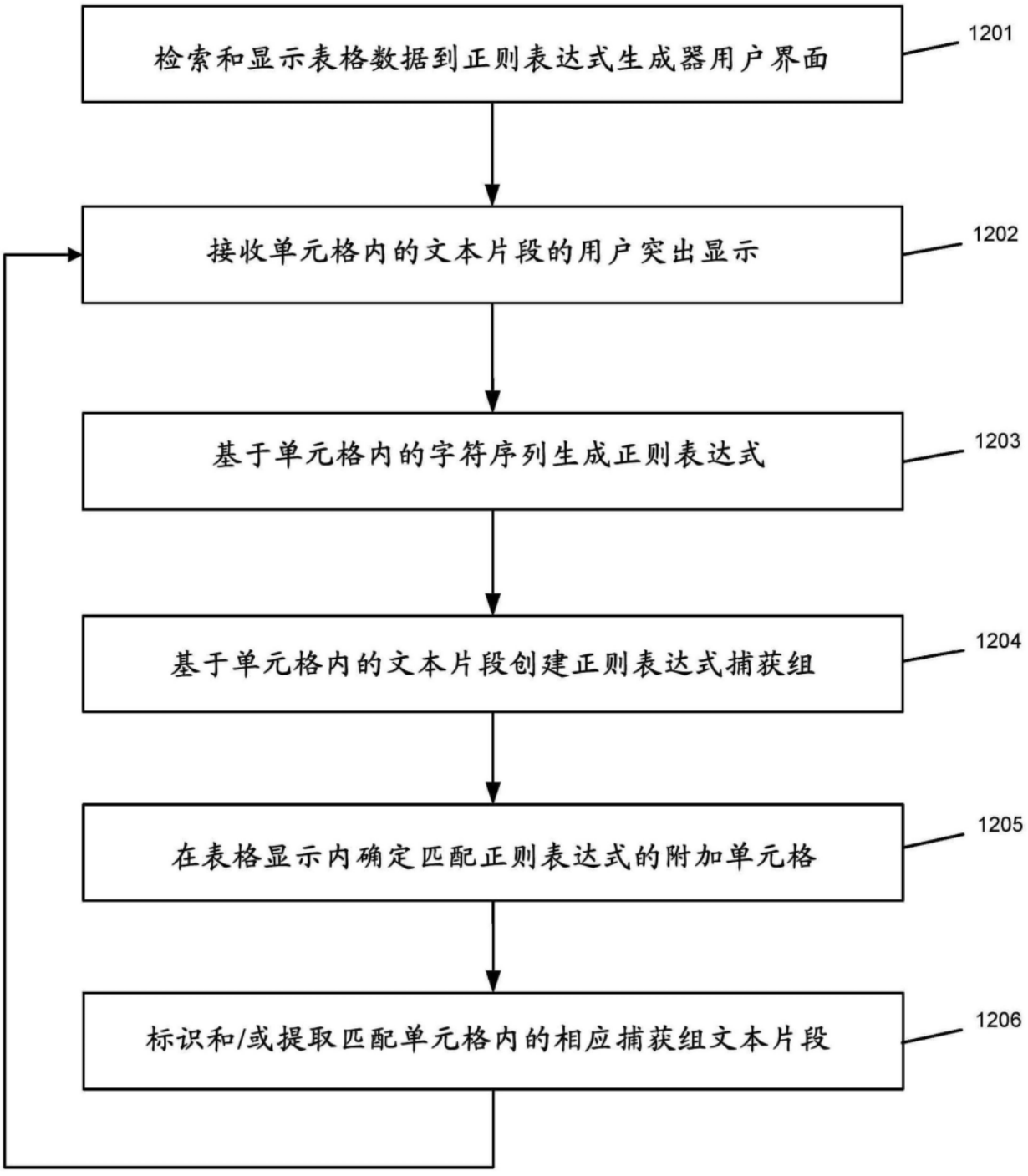

检索和显示表格数据到正则表达式生成器用户界面
1201
接收单元格内的文本片段的用户突出显示
1202
基于单元格内的字符序列生成正则表达式
1203
基于单元格内的文本片段创建正则表达式捕获组
1204
在表格显示内确定匹配正则表达式的附加单元格
1205
标识和/或提取匹配单元格内的相应捕获组文本片段
1206

图12

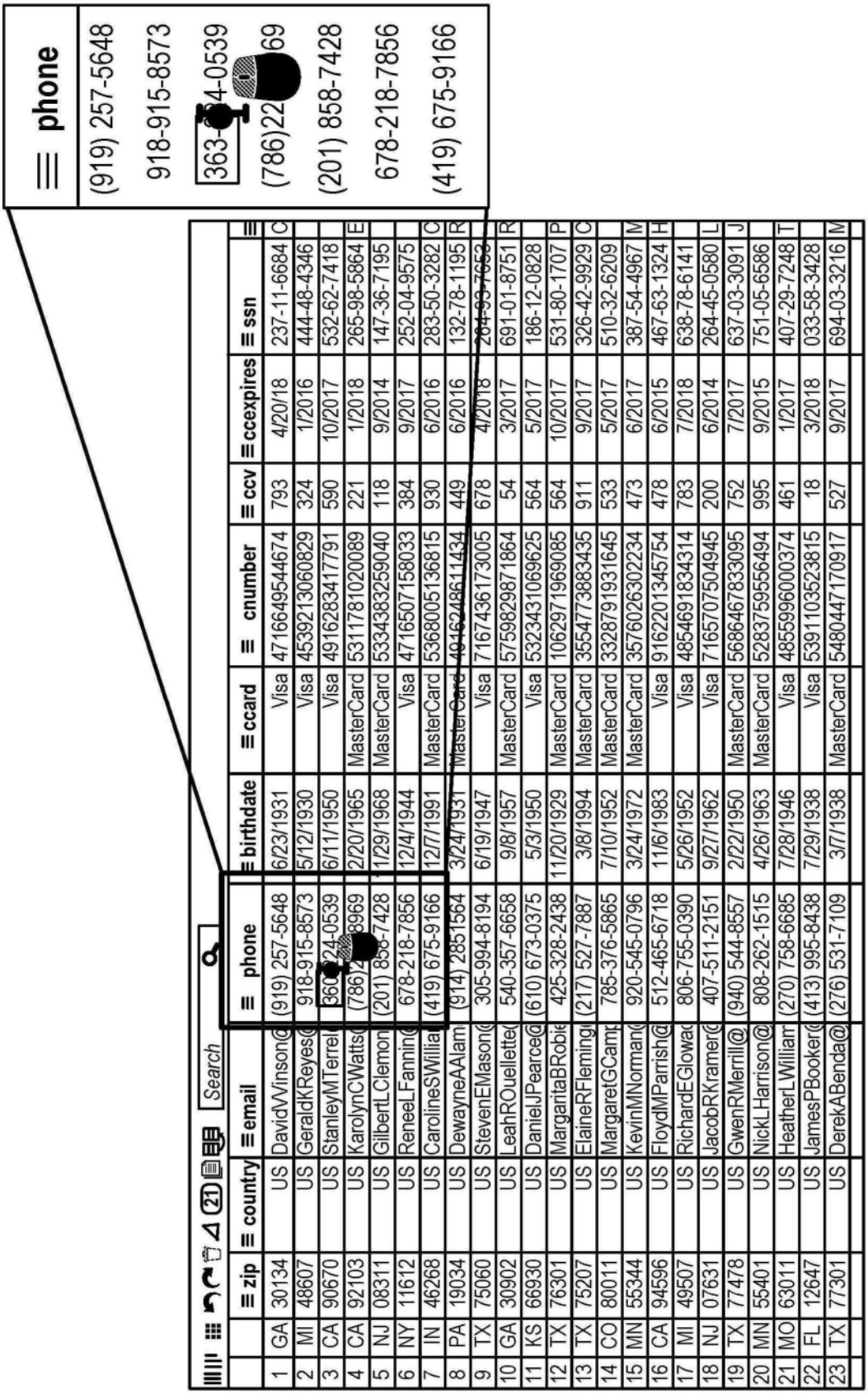

phone
(919) 257-5648
918-915-8573
(786)22 69
(201) 858-7428
678-218-7856
(419) 675-9166
Search
zip
country
email
phone
birthdate
ccard
cnumber
ccv
ccexpires
ssn
1 GA 30134 US DavidWVinson@ (919) 257-5648 6/23/1931 Visa 4716649544674 793 4/20/18 237-11-6684
2 MI 48607 US GeraldKReyes@ 918-915-8573 5/12/1930 Visa 4539213060829 324 1/2016 444-48-4346
3 CA 90670 US StanleyMTerrel 360 24-0539 6/11/1950 Visa 4916283417791 590 10/2017 532-62-7418
4 CA 92103 US KarolynCWatts((786)2 8969 2/20/1965 MasterCard 5311781020089 221 1/2018 265-98-5864
5 NJ 08311 US GilbertLClemon (201) 858-7428 1/29/1968 MasterCard 5334383259040 118 9/2014 147-36-7195
6 NY 11612 US ReneeLFannin(678-218-7856 12/4/1944 Visa 4716507158033 384 9/2017 252-04-9575
7 IN 46268 US CarolineSWillia (419) 675-9166 12/7/1991 MasterCard 5368005136815 930 6/2016 283-50-3282
8 PA 19034 US DewayneAAlam (914) 285 1564 3/24/1931 MasterCard 4046248611434 449 6/2016 132-78-1195
9 TX 75060 US StevenEMason@ 305-994-8194 6/19/1947 Visa 7167436173005 678 4/2018 284-93-7663
10 GA 30902 US LeahROuellette(540-357-6658 9/8/1957 MasterCard 5759829871864 54 3/2017 691-01-8751
11 KS 66930 US DanielJPearce@ (610) 673-0375 5/3/1950 Visa 5323431069625 564 5/2017 186-12-0828
12 TX 76301 US MargaritaBRobi 425-328-2438 11/20/1929 MasterCard 1062971969085 564 10/2017 531-80-1707
13 TX 75207 US ElaineRFleming 217) 527-7887 3/8/1994 MasterCard 3554773883435 911 9/2017 326-42-9929
14 CO 80011 US MargaretGCamp 785-376-5865 7/10/1952 MasterCard 3328791931645 533 5/2017 510-32-6209
15 MN 55344 US KevinMNorman(920-545-0796 3/24/1972 MasterCard 3576026302234 473 6/2017 387-54-4967
16 CA 94596 US FloydMParrish@ 512-465-6718 11/6/1983 Visa 9162201345754 478 6/2015 467-63-1324
17 MI 49507 US RichardEGlowac 806-755-0390 5/26/1952 Visa 4854691834314 783 7/2018 638-78-6141
18 NJ 07631 US JacobRKramer@ 407-511-2151 9/27/1962 Visa 7165707504945 200 6/2014 264-45-0580
19 TX 77478 US GwenRMerrill@ (940) 544-8557 2/22/1950 MasterCard 5686467833095 752 7/2017 637-03-3091
20 MN 55401 US NickLHarrison@ 808-262-1515 4/26/1963 MasterCard 5283759556494 995 9/2015 751-05-6586
21 MO 63011 US HeatherLWilliam (270) 758-6685 7/28/1946 Visa 4855996000374 461 1/2017 407-29-7248
22 FL 12647 US JamesPBooker((413) 995-8438 7/29/1938 Visa 5391103523815 18 3/2018 033-58-3428
23 TX 77301 US DerekABenda@ (276) 531-7109 3/7/1938 MasterCard 5480447170917 527 9/2017 694-03-3216

图13

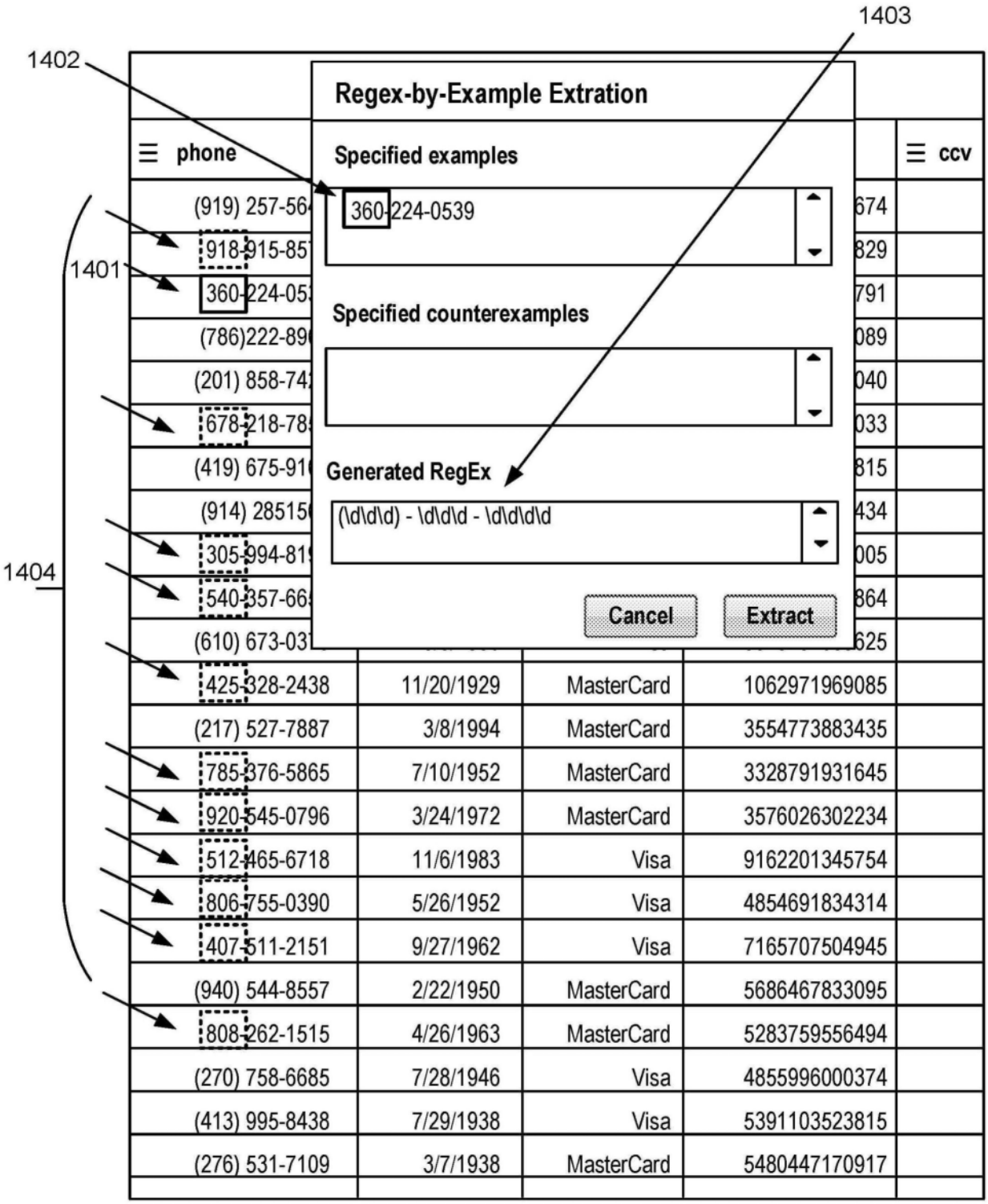

1403
1402
1401
1404
Regex-by-Example Extration
Specified examples
360-224-0539
Specified counterexamples
Generated RegEx
(\d\d\d) - \d\d\d - \d\d\d\d
Cancel
Extract
phone
ccv
(919) 257-56
918-915-85
360-224-05
(786)222-89
(201) 858-74
678-218-78
(419) 675-91
(914) 28515
305-994-81
540-357-66
(610) 673-03
425-328-2438 11/20/1929 MasterCard 1062971969085
(217) 527-7887 3/8/1994 MasterCard 3554773883435
785-376-5865 7/10/1952 MasterCard 3328791931645
920-545-0796 3/24/1972 MasterCard 3576026302234
512-465-6718 11/6/1983 Visa 9162201345754
806-755-0390 5/26/1952 Visa 4854691834314
407-511-2151 9/27/1962 Visa 7165707504945
(940) 544-8557 2/22/1950 MasterCard 5686467833095
808-262-1515 4/26/1963 MasterCard 5283759556494
(270) 758-6685 7/28/1946 Visa 4855996000374
(413) 995-8438 7/29/1938 Visa 5391103523815
(276) 531-7109 3/7/1938 MasterCard 5480447170917

图14

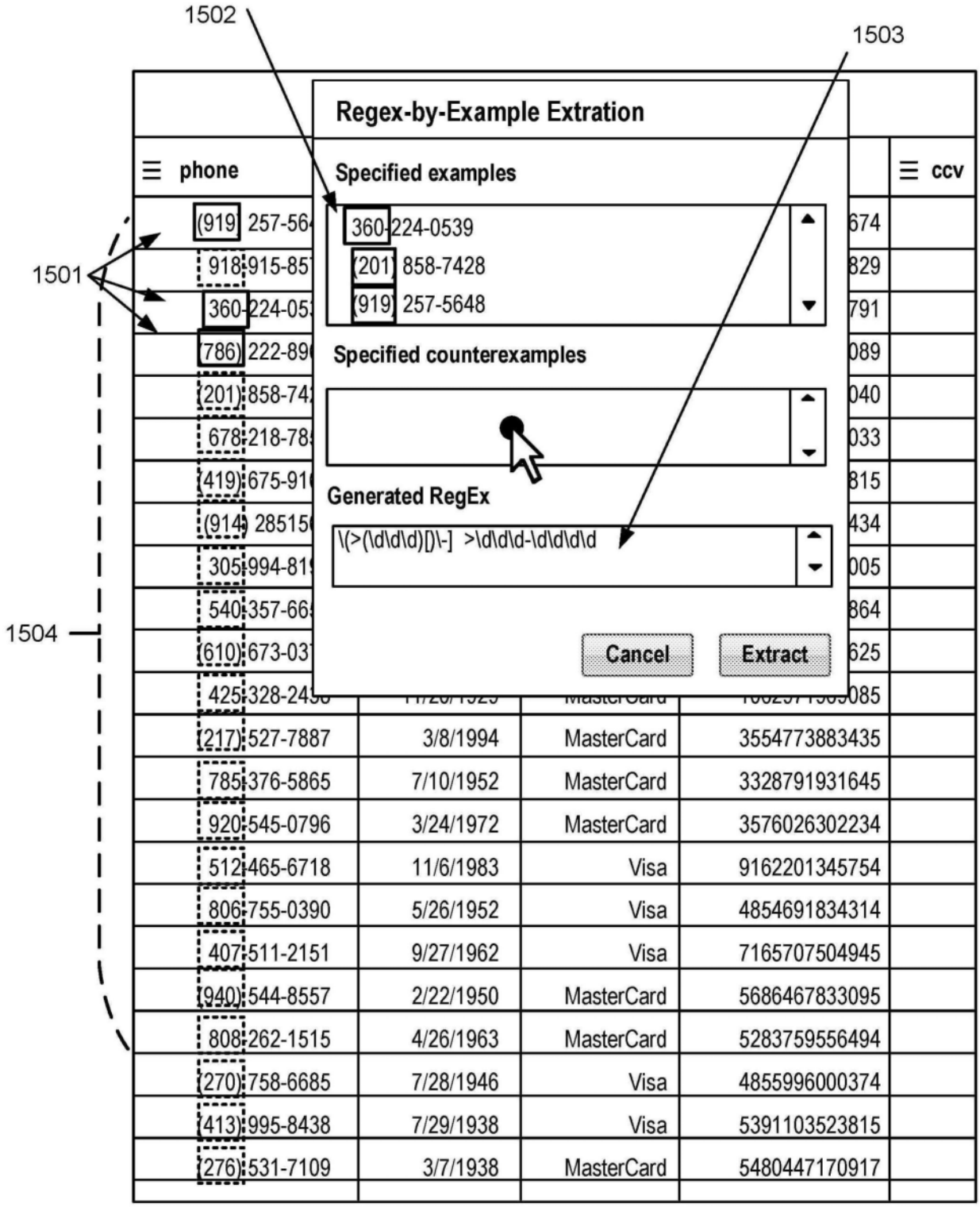

1502
1503
1501
1504
Regex-by-Example Extration
Specified examples
360-224-0539
(201) 858-7428
(919) 257-5648
Specified counterexamples
Generated RegEx
\(>(\d\d\d)[)\-] >\d\d\d-\d\d\d\d
Cancel
Extract
phone
ccv
(919) 257-56
918) 915-85
360-224-05
(786) 222-89
(201) 858-74
678) 218-78
(419) 675-91
(914) 28515
305) 994-81
540) 357-66
(610) 673-03
425) 328-24
(217) 527-7887 3/8/1994 MasterCard 3554773883435
785) 376-5865 7/10/1952 MasterCard 3328791931645
920) 545-0796 3/24/1972 MasterCard 3576026302234
512) 465-6718 11/6/1983 Visa 9162201345754
806) 755-0390 5/26/1952 Visa 4854691834314
407) 511-2151 9/27/1962 Visa 7165707504945
(940) 544-8557 2/22/1950 MasterCard 5686467833095
808) 262-1515 4/26/1963 MasterCard 5283759556494
(270) 758-6685 7/28/1946 Visa 4855996000374
(413) 995-8438 7/29/1938 Visa 5391103523815
(276) 531-7109 3/7/1938 MasterCard 5480447170917

图15

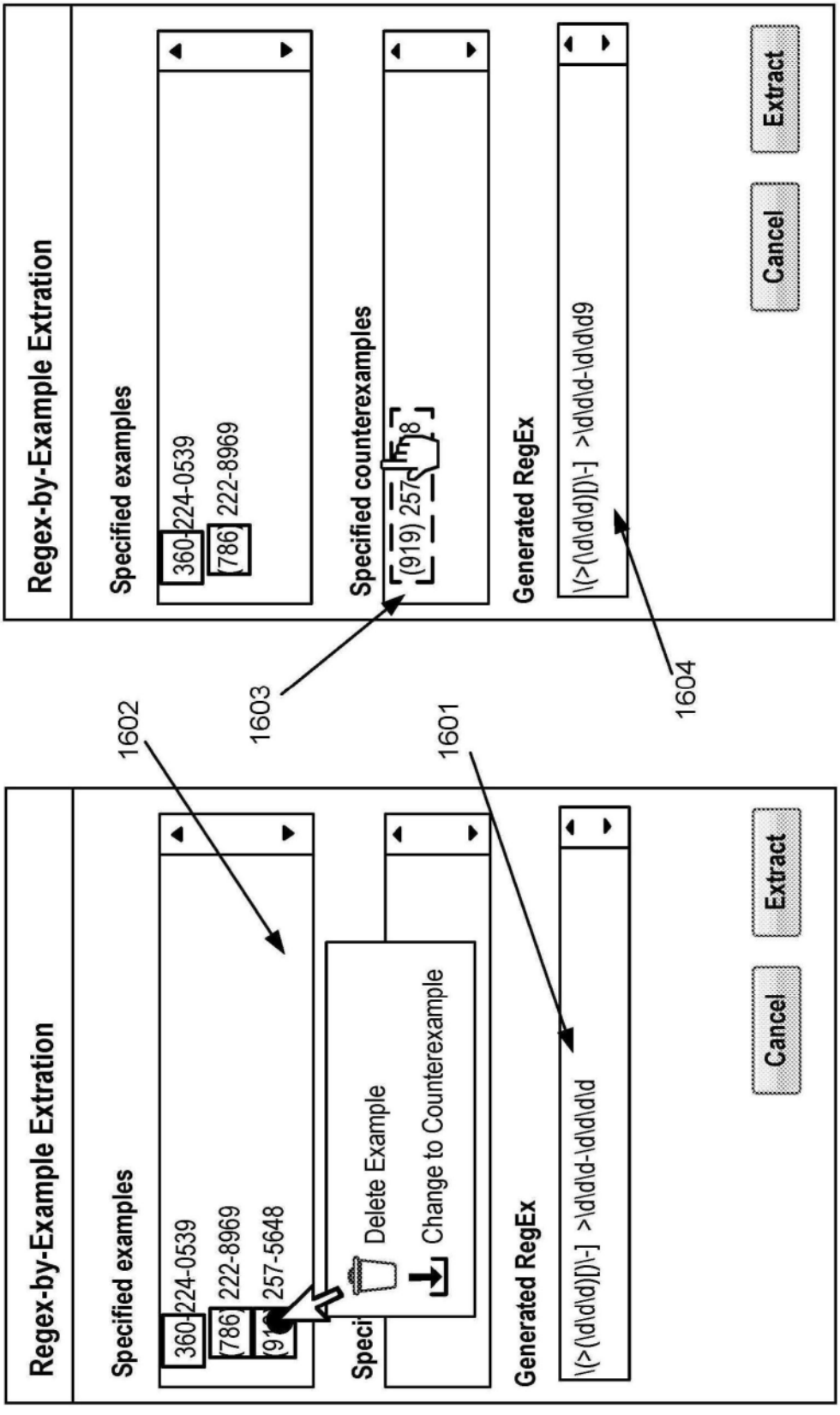

Regex-by-Example Extration
Specified examples
360-224-0539
786-222-8969
257-5648
Delete Example
Change to Counterexample
Specified counterexamples
Generated RegEx
\(>(\d\d\d)[)\-] >\d\d\d-\d\d\d\d
Cancel
Extract
1601
1602
1603
1604

图16A

图16B

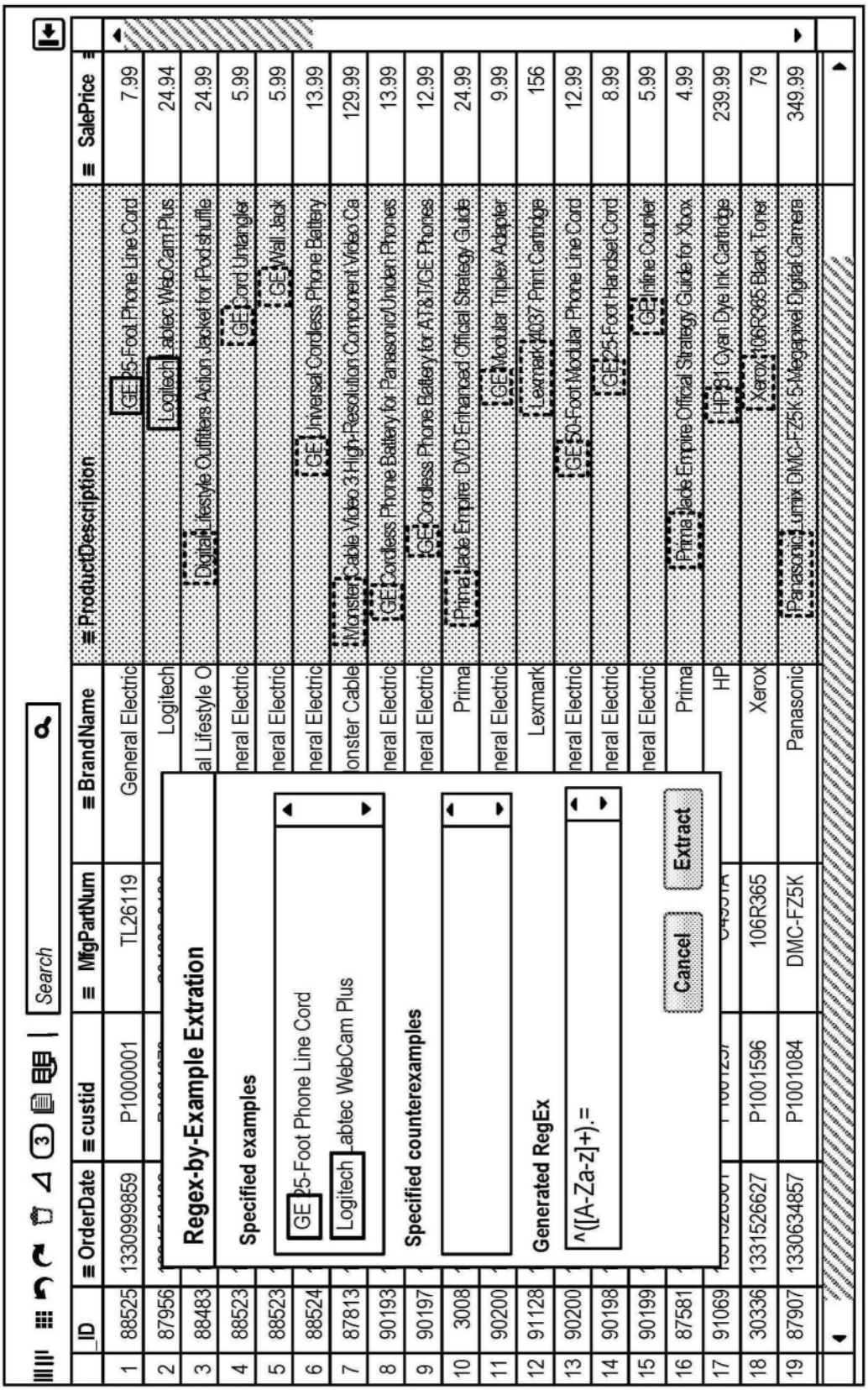

Search
ID
OrderDate
custid
MfgPartNum
BrandName
ProductDescription
SalePrice
1 88525 1330999859 P1000001 TL26119 General Electric GE 25-Foot Phone Line Cord 7.99
2 87956 Logitech Logitech Labtec WebCam Plus 24.94
3 88483 al Lifestyle O Digital Lifestyle Outfitters Action Jacket for iPod shuffle 24.99
4 88523 neral Electric GE Cord Untangler 5.99
5 88523 neral Electric GE Wall Jack 5.99
6 88524 neral Electric GE Universal Cordless Phone Battery 13.99
7 87813 onster Cable Monster Cable Video 3 High-Resolution Component Video Ca 129.99
8 90193 neral Electric GE Cordless Phone Battery for Panasonic/Uniden Phones 13.99
9 90197 neral Electric GE Cordless Phone Battery for AT&T/GE Phones 12.99
10 3008 Prima Prima Jade Empire: DVD Enhanced Official Strategy Guide 24.99
11 90200 neral Electric GE Modular Triplex Adapter 9.99
12 91128 Lexmark Lexmark 1037 Print Cartridge 156
13 90200 neral Electric GE 50-Foot Modular Phone Line Cord 12.99
14 90198 neral Electric GE 25-Foot Handset Cord 8.99
15 90199 neral Electric GP Inline Coupler 5.99
16 87581 Prima Prima Jade Empire Official Strategy Guide for Xbox 4.99
17 91069 HP HP 81 Cyan Dye Ink Cartridge 239.99
18 30336 1331526627 P1001596 106R365 Xerox Xerox 106R365 Black Toner 79
19 87907 1330634857 P1001084 DMC-FZ5K Panasonic Panasonic Lumix DMC-FZ5K 5-Megapixel Digital Camera 349.99
Regex-by-Example Extration
Specified examples
GE 25-Foot Phone Line Cord
Logitech Labtec WebCam Plus
Specified counterexamples
Generated RegEx
^([A-Za-z]+).=
Cancel
Extract

图17

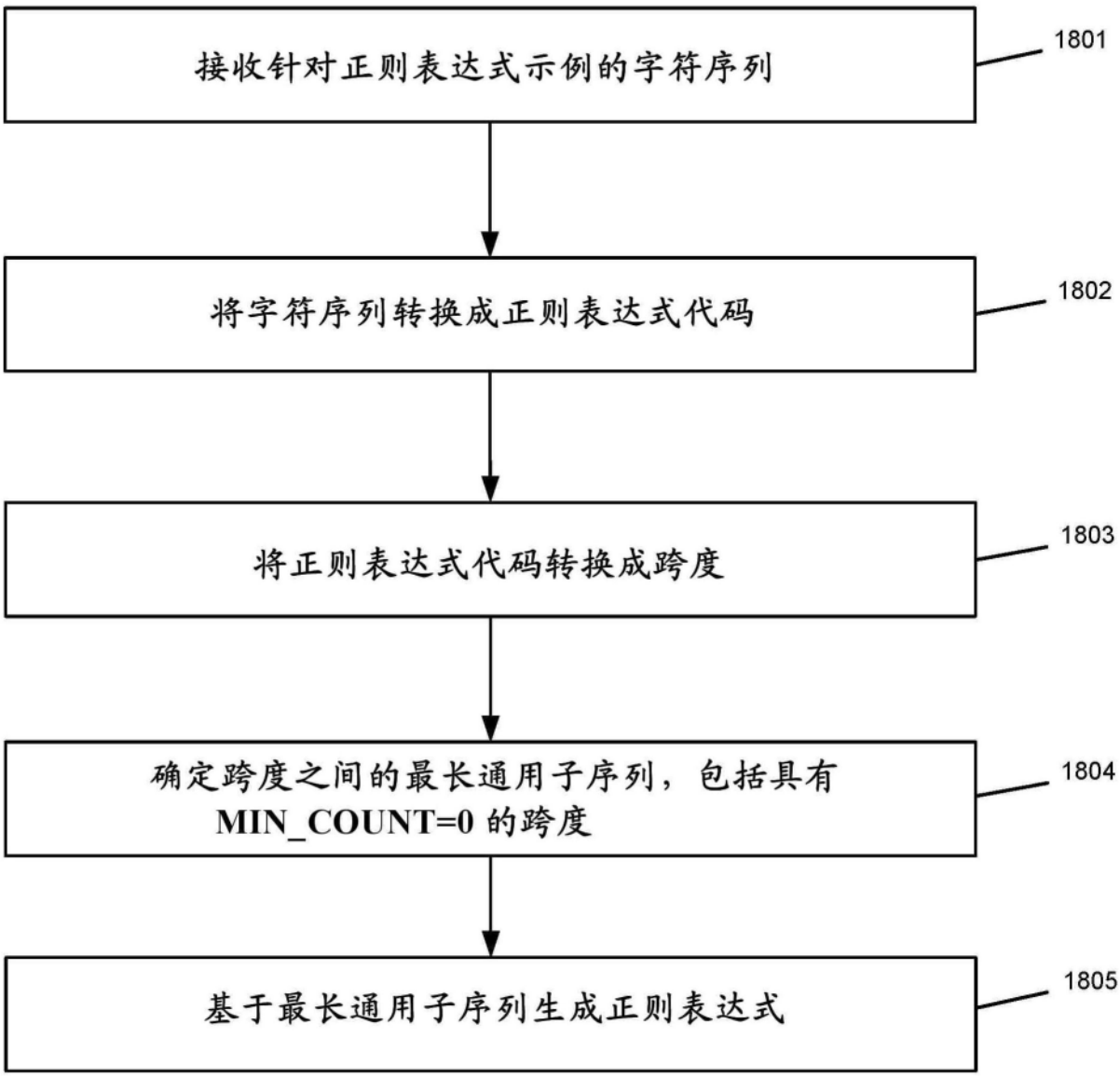

接收针对正则表达式示例的字符序列
1801
将字符序列转换成正则表达式代码
1802
将正则表达式代码转换成跨度
1803
确定跨度之间的最长通用子序列，包括具有MIN_COUNT=0 的跨度
1804
基于最长通用子序列生成正则表达式
1805

图18

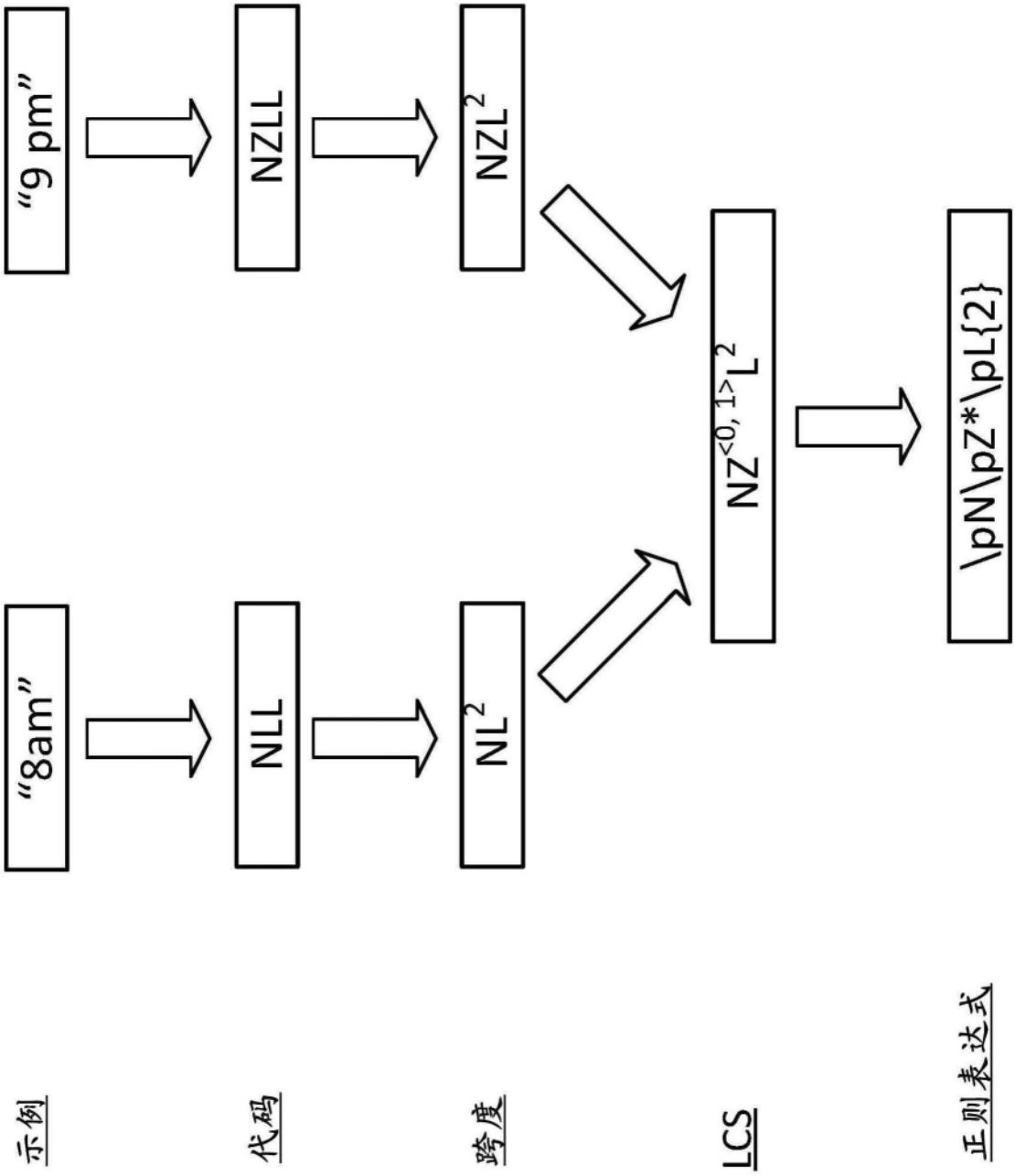


图19

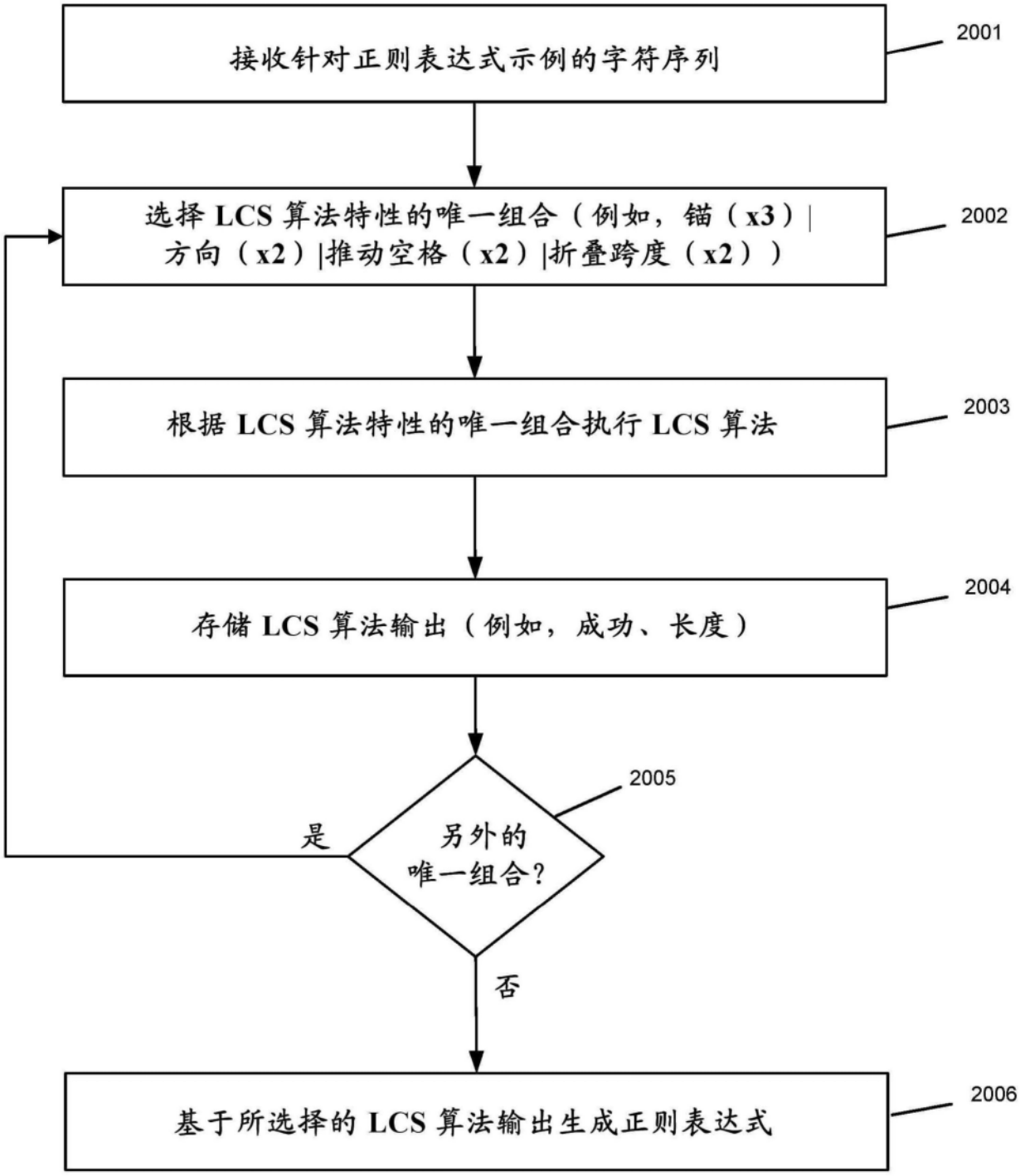

接收针对正则表达式示例的字符序列
2001
选择 LCS 算法特性的唯一组合（例如，锚（x3）|方向（x2）|推动空格（x2）|折叠跨度（x2））
2002
根据 LCS 算法特性的唯一组合执行 LCS 算法
2003
存储 LCS 算法输出（例如，成功、长度）
2004
另外的唯一组合?
2005
是
否
基于所选择的 LCS 算法输出生成正则表达式
2006

图20

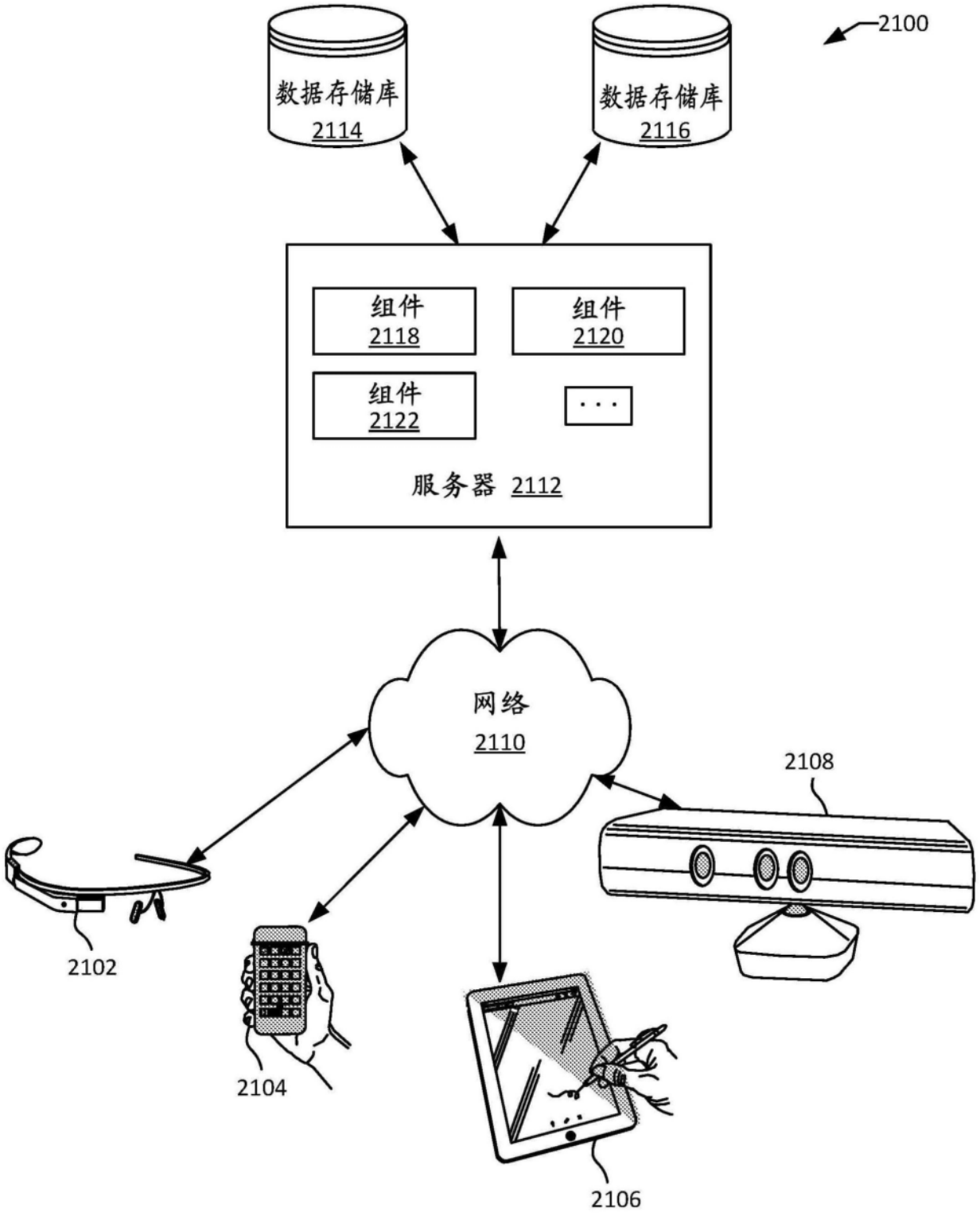

2100
数据存储库
2114
数据存储库
2116
组件
2118
组件
2120
组件
2122
服务器 2112
网络
2110
2108
2102
2104
2106

图21

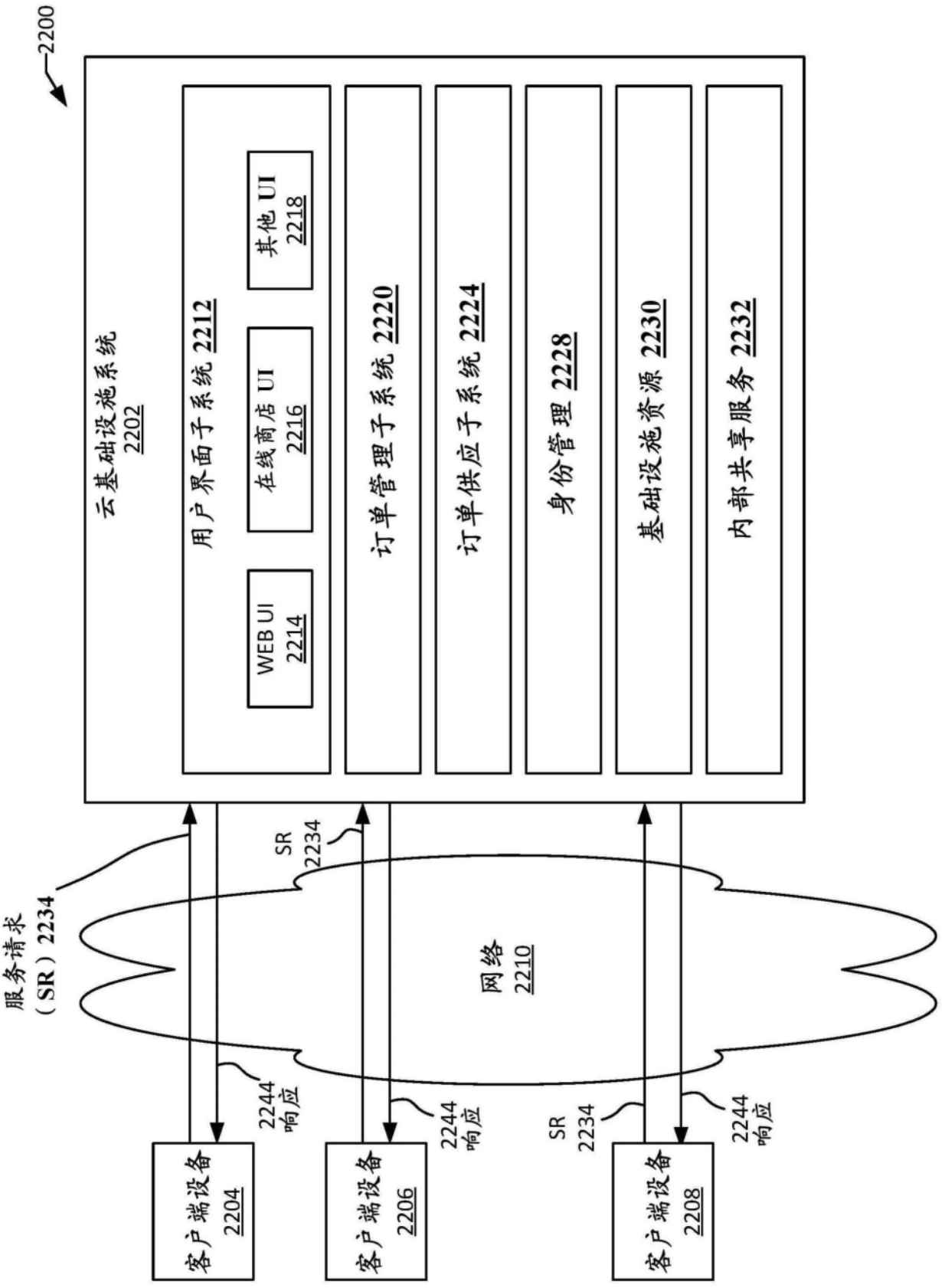

2200
云基础设施系统 2202
用户界面子系统 2212
WEB UI 2214
在线商店 UI 2216
其他 UI 2218
订单管理子系统 2220
订单供应子系统 2224
身份管理 2228
基础设施资源 2230
内部共享服务 2232
服务请求（SR）2234
SR 2234
网络 2210
2244 响应
2244 响应
SR 2234
2244 响应
客户端设备 2204
客户端设备 2206
客户端设备 2208

图22

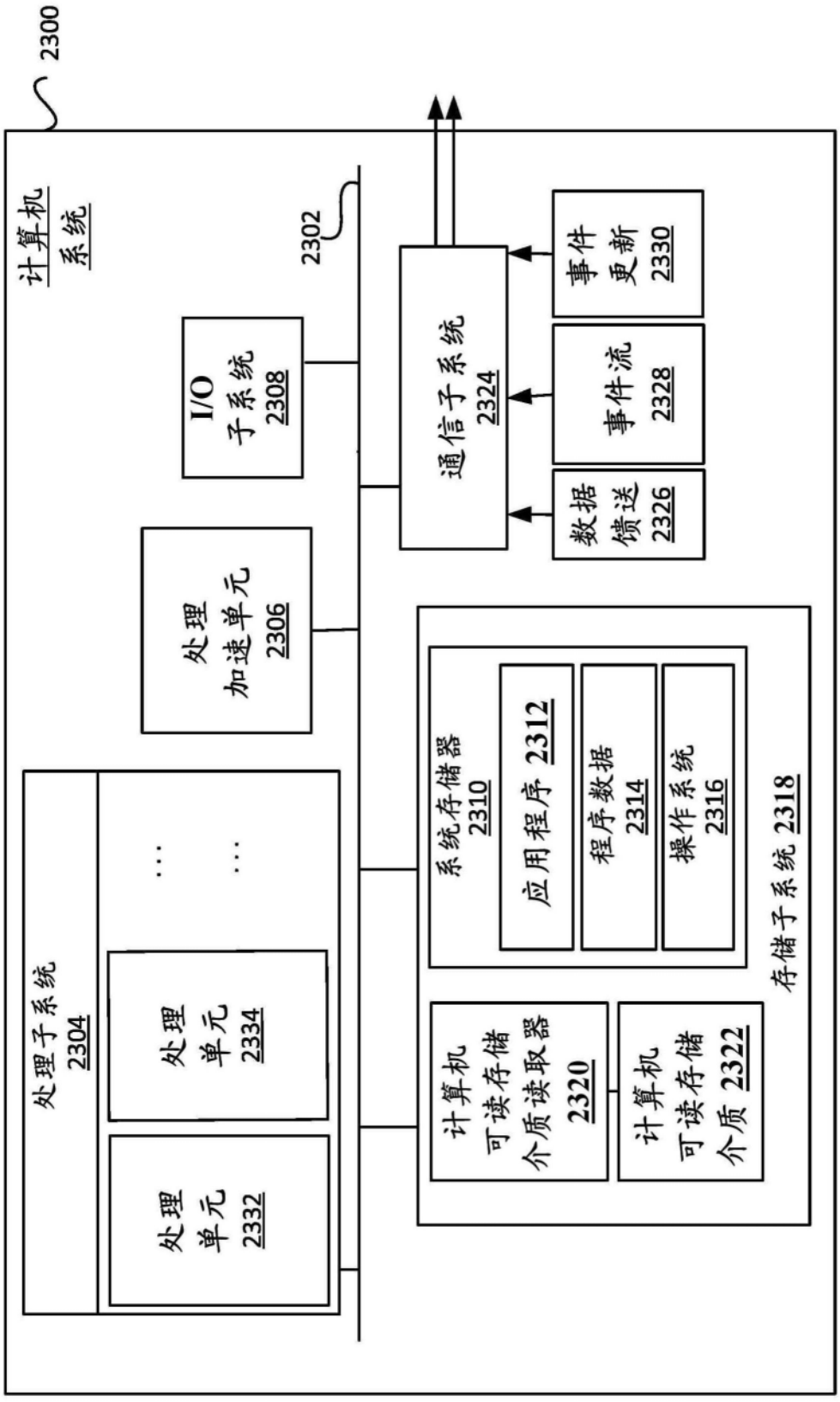

2300
计算机系统
2302
I/O子系统 2308
处理加速单元 2306
通信子系统 2324
事件更新 2330
事件流 2328
数据馈送 2326
处理子系统 2304
处理单元 2332
处理单元 2334
...
...
系统存储器 2310
应用程序 2312
程序数据 2314
操作系统 2316
计算机可读存储介质读取器 2320
计算机可读存储介质 2322
存储子系统 2318

图23