(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 112262368 A
(43)申请公布日 2021.01.22

(21)申请号 201980018860.9
(22)申请日 2019.03.05
(30)优先权数据
15/919,209 2018.03.13 US
(85)PCT国际申请进入国家阶段日
2020.09.11
(86)PCT国际申请的申请数据
PCT/US2019/020853 2019.03.05
(87)PCT国际申请的公布数据
WO2019/177819 EN 2019.09.19
(71)申请人 微软技术许可有限责任公司
地址 美国华盛顿州
(72)发明人 A·H·亚瓦达拉 王淼森
R·怀特 苏煜
(74)专利代理机构 北京市金杜律师事务所 11256
代理人 黄倩
(51)Int.Cl.
G06F 8/30(2018.01)
G06F 8/38(2018.01)

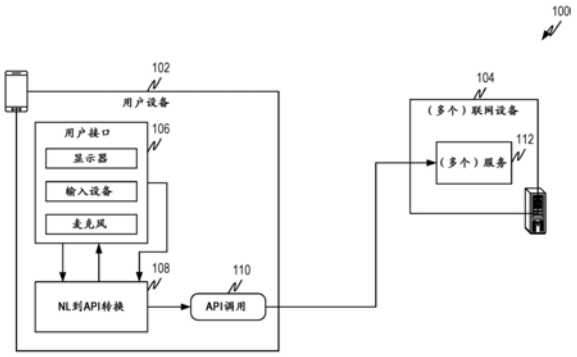
权利要求书2页 说明书21页 附图12页

(54)发明名称

自然语言到API转换

(57)摘要

代表性实施例公开了将自然语言输入映射到应用编程接口(API)调用的机制。自然语言输入被首先映射到API框架,其是没有任何API调用格式化的API调用的表示。从自然语言输入到API框架的映射使用经训练的序列到序列神经模型来执行。序列到序列神经模型被分解成被称为模块的小预测单元。每个模块在预测预定义种类的序列输出方面高度专业化。模块的输出可以被显示在允许用户添加、移除和/或修改个体模块的输出的交互式用户界面中。用户输入可以被用作另外的训练数据。使用确定性映射,API框架被映射到API调用。



CN 112262368 A

1.一种计算机实现的方法,包括:

接收自然语言话语;

将所述自然语言话语提交到包括序列到序列神经模型的经训练的机器学习模型,所述序列到序列神经模型包括编码器和多个解码器,所述多个解码器中的每个解码器被耦合到所述编码器,并且被训练为识别从所述编码器输出的一个或多个标记,并且将所述一个或多个标记映射到API框架的一个或多个项;

从所述经训练的机器学习模型接收多个项,每个项从不同的编码器被接收;

将所述多个项汇编到所述API框架中,所述API框架表示所述自然语言话语与最终API格式之间的中间格式;

将所述API框架映射到所述最终API格式;以及

使用所述最终API格式来发出对所述API的调用。

2.根据权利要求1所述的方法,其中所述编码器包括递归神经网络。

3.根据权利要求1所述的方法,其中每个解码器包括注意力递归神经网络。

4.根据权利要求1所述的方法,还包括被耦合到所述编码器的控制器,所述控制器接收所述编码器的输出,并产生包括所述多个解码器的布局,并激活所述布局中的所述解码器中的每个解码器。

5.根据权利要求1所述的方法,还包括:

经由用户界面呈现每个项,所述用户界面包括:

区域,用于呈现每个项以及指示所述项意指什么的相关联的指示;

控件,被激活时移除相关联的项;

控件,被激活时添加新的项;

经由所述用户界面接收指示对所述多个项的任何改变完成的输入。

6.根据权利要求5所述的方法,还包括:

聚合执行数据,所述执行数据包括:

所述自然语言话语;

经汇编的所述API框架;以及

经由所述用户界面对多个项做出的任何改变。

7.根据权利要求1所述的方法,其中将所述API框架映射到所述最终API格式使用多个规则来完成,所述多个规则将所述API框架的多个项确定性地映射到所述最终API格式中的多个项。

8.根据权利要求1所述的方法,其中所述编码器和所述多个解码器使用监督式学习过程被训练,所述监督式学习过程利用经注释的自然语言话语数据,所述经注释的自然语言话语数据包括:

多个训练自然语言话语;以及

针对所述训练自然语言话语中的每个训练自然语言话语的API框架。

9.根据权利要求1所述的方法,还包括:使用利用交互式用户界面收集的数据来更新对所述序列到序列神经模型的训练,所述交互式用户界面呈现从所述多个解码器输出的多个项并允许:

项在将多个项汇编到所述API框架中之前被移除;

项在将多个项汇编到所述API框架中之前被添加;以及

项在将多个项汇编到所述API框架中之前被修改。

10.一种系统,包括:

处理器和具有可执行指令的设备存储介质,所述可执行指令在由所述处理器执行时使所述系统执行包括以下的操作:

接收自然语言话语;

将所述自然语言话语提交到包括序列到序列神经模型的经训练的机器学习模型,所述序列到序列神经模型包括编码器和多个解码器,所述多个解码器中的每个解码器被耦合到所述编码器,并被训练为识别从所述编码器输出的一个或多个标记,并将所述一个或多个标记映射到API框架的一个或多个项;

从所述经训练的机器学习模型接收多个项,每个项从不同的编码器被接收;

将所述多个项汇编到所述API框架中,所述API框架表示所述自然语言话语与最终API格式之间的中间格式;

将所述API框架映射到所述最终API格式;以及

使用所述最终API格式来发出对所述API的调用。

11.根据权利要求10所述的系统,其中所述编码器包括递归神经网络。

12.根据权利要求10所述的系统,其中每个解码器包括注意力递归神经网络。

13.根据权利要求10所述的系统,还包括被耦合到所述编码器的控制器,所述控制器:

接收所述编码器的输出;

产生包括所述多个解码器的布局;以及

激活所述布局中的所述解码器中的每个解码器。

14.根据权利要求10所述的系统,还包括:

经由用户界面呈现每个项,所述用户界面包括:

区域,用于呈现每个项以及指示所述项意指什么的相关联的指示;

控件,被激活时移除相关联的项;

控件,被激活时添加新的项;

经由所述用户界面接收指示对所述多个项的任何改变完成的输入。

15.根据权利要求14所述的系统,还包括:

聚合执行数据,所述执行数据包括:

所述自然语言话语;

经汇编的所述API框架;以及

经由所述用户界面对多个项做出的任何改变;以及

将所聚合的数据发送到系统以被用于更新对所述序列到序列神经模型的训练。

自然语言到API转换

技术领域

[0001] 本申请总体上涉及自然语言处理。更具体地，在一些方面中，本申请涉及使用经训练的机器学习模型来将自然语言输入转换成应用编程接口调用。

背景技术

[0002] 计算的快速增长的普遍性对下一代人机接口提出了极大的需求。自然语言接口被广泛地认为是一个有前途的方向。用户想要使用自然语言措词(无论是说出的、键入的还是以其他方式输入的)控制各种各样的设备、程序、应用等。

[0003] 本实施例在这种背景下产生。

附图说明

[0004] 图1图示了根据本公开的一些方面的代表性系统架构。

[0005] 图2图示了根据本公开的一些方面的另一代表性系统架构。

[0006] 图3图示了根据本公开的一些方面的用于将自然语言输入转换成应用编程接口(API)调用的代表性流程图。

[0007] 图4图示了根据本公开的一些方面的用于使用机器学习模型将自然语言输入转换成API调用的代表性流程图。

[0008] 图5图示了根据本公开的一些方面的用于使用机器学习模型将自然语言输入转换成API框架的代表性流程图。

[0009] 图6图示了根据本公开的一些方面的用于使用机器学习模型将自然语言输入转换成API框架的代表性流程图。

[0010] 图7图示了用于将API框架映射到API调用的代表性流程图。

[0011] 图8图示了示出根据本公开的一些方面的用于使用机器学习模型将自然语言输入转换成API调用的交互式用户界面的代表性示图。

[0012] 图9图示了来自根据本公开的一些方面的用于使用机器学习模型将自然语言输入转换成API调用的代表性交互式用户界面的反馈。

[0013] 图10图示了根据本公开的一些方面的具有机器学习模型训练的代表性系统架构。

[0014] 图11图示了根据本公开的一些方面的用于机器学习模型的代表性训练过程。

[0015] 图12图示了适合于实现本文公开的系统和其他方面或适合于执行本文公开的方法的代表性机器架构。

具体实施方式

[0016] 下面的描述包括例示说明性实施例的说明性系统、方法、用户接口、技术、指令序列、以及计算机器程序产品。在下面的描述中，为了解释的目的，阐述了许多具体细节以提供对本发明主题的各种实施例的理解。然而，对于本领域技术人员将显而易见的是，可以在没有这些具体细节的情况下实践本发明主题的实施例。总体上，未详细示出公知的指令实

例、协议、结构以及技术。

[0017] 概述

[0018] 以下概述被提供从而以简化的形式介绍下面在说明中进一步描述的一系列概念。本概述不旨在标识要求保护的主题的关键特征或必要特征，也不旨在用于限制要求保护的主题的范围。其唯一目的是以简化的形式呈现一些构思作为对稍后呈现的更详细的描述的前序。

[0019] 随着自然语言变成到计算设备中的更常见的接口方法，越来越多的用户想要利用自然语言来接口连接并控制用户可以利用计算设备完成的几乎所有事情。遗憾的是，许多(如果不是大多数的话)应用、程序、服务、数字助理、设备(诸如所谓的“物联网”)以及用户在计算设备上利用的其他工具(在本文统称为程序)不是启用自然语言(natural language enabled)的。例如，一些数据库和程序期望特定格式和/或语言的输入。即使对于启用自然语言的那些程序，程序识别自然语言并转化成用户期望的一个或多个动作的能力可以是不均匀的并且在各程序之间变化。

[0020] 尽管到程序的自然语言接口可以缺失或不均匀，但是几乎每个程序都显露(expose)允许其他编程实体与程序接口连接并通过该程序完成期望活动的一个或多个应用编程接口(API)。本公开的实施例将自然语言输入转换成合适的API调用。这样的转换允许在用户利用程序来完成期望任务时的针对用户的统一体验。这还允许通过自然语言来控制未启用自然语言的程序。

[0021] 自然语言接口的核心挑战在于将来自用户的自然语言话语(命令)映射到形式意义表示，诸如API。传统方法已经依赖于基于规则的模型，其使用一组规则来将话语映射到API调用中。然而，这样的模型通常是脆弱的并且不能处理自然语言的灵活性和模糊性。另外，这样的模型常常是手工编码的并且如果用户使用新话语，或者如果开发者需要添加新话语或API，那么开发者必须将该新话语和相关联的API添加到模型中。

[0022] 统计方法也已经被尝试，其中机器学习模型利用大量人工设计的特征来训练。然而，这是非常耗时的并且要求针对每个自然语言接口和针对每个应用域的设计，其可以限制它们的适用性。另外，这样的方法通常导致二元决策，模型正确地解读了用户想要什么或模型未正确地解读用户想要什么。这向用户呈现了“黑盒”问题，因为如果模型未正确地解读用户想要什么，那么用户不能校正所识别的内容的不正确部分，并且用户必须简单地再次尝试。在语音输入方面这可能尤其令人沮丧，其中用户一遍又一遍地说出相同短语并且设备一遍又一遍地以不想要的解释做出响应。这通常导致用户简单地放弃任何尝试并以不同的方式来处理问题。

[0023] 本文公开的方法和实施例可以被应用于将自然语言转换成任何一组或任何类型的API，包括本地API、远程过程调用、Web API等。自然语言到API转换借助于经训练的机器学习模型来发生，该模型将序列到序列机器学习模型分解成被称为模块的较小预测单元。每个模块在预测预定义种类的序列输出方面高度专业化。一个大的益处在于，个体预测模块的输出可以被容易地映射到用户可理解的概念中。如果有必要的话，在一些实施例中，这些可以被呈现给用户以允许用户验证和校正。这避免传统方法的“黑盒”问题。这还允许提高与设备的自然语言交互的效率和准确性的新的用户交互机制。另外，模型能够以先前未通过任何自然语言到API转换方法实现的方式来执行。

[0024] 因此,所公开的实施例通过实现用户与设备之间的被极大改进的交互来改进机器的操作。所公开的实施例还通过实现与未启用自然语言的程序的自然语言交互,从而提供与现有程序的用户交互的新模式来改进机器的操作。所公开的实施例还通过实现对所接收的自然语言输入的自然语言解释的用户交互和校正,来改进机器的操作。这还通过针对本文使用的机器学习模型的更快训练和更好准确性来改进机器的操作。

[0025] 说明

[0026] 图1图示了根据本公开的一些方面的代表性系统架构100。在该架构中,自然语言(NL)到API转换发生在用户设备102上。用户设备102经由用户接口106接收一个或多个NL话语。NL话语可以包括短语、句子或其他NL输入。NL输入可以以文本、语音或其他NL输入的形式。用户接口106可以包括显示器(诸如触摸屏或其他显示器)、输入设备,诸如键盘、触摸屏上显示的虚拟键盘等,以及附加输入机构(诸如用于语音输入的麦克风)。用户接口106可以还包括扬声器或其他音频输出设备,以使得使用音频的NL到API转换交互性可以如本文所描述的被实现。用户设备102的其他方面下面在图11中进行讨论。

[0027] NL话语被发送到NL到API转换过程108。NL到API转换108包括如下面所描述的经训练的机器学习模型。在一个实施例中,经训练的机器学习模型包括序列到序列模型,其中,解码器方面包括多个专业化的解码器,每个被训练为识别来自编码器的预定义种类的序列输出。这些专业化的解码器在本文被称为模块。机器学习模型在下面更详细地进行描述。NL话语到API的转换在下面更完整地进行描述。

[0028] 因为解码器模块被训练为识别预定义种类的序列输出,所以它们映射到(多个)NL话语的部分并且因此所识别的序列的种类可以被容易地描述给用户。这为用户交互性呈现避免上述黑盒问题的机会。所识别的NL话语可以经由用户接口以文本、声音和/或以其他格式和/或其组合被呈现给用户。用户可以然后校正已经被不正确地解读的话语的任何部分,并且然后可以提交经校正的所识别的话语,并且NL到API转换108可以然后创建适当的API调用110。交互性在下面进一步讨论。

[0029] 这样的交互性不需要是本公开的所有实施例的部分。在具有如本文所描述的某些益处的一些实施例中,这仅是一个选项。

[0030] NL到API转换108可以经由硬件(诸如在神经网络芯片中)实现,该硬件被设计为执行经训练的机器学习模型,或者经由(诸如其中可执行指令经由一个或多个硬件处理器执行)硬件和软件的组合来实现,或两种方式都可以。

[0031] API调用110是可以然后由用户设备、由用户设备上的应用或程序、或由另一编程和/或硬件实体使用以执行API调用的格式化API。例如,如果API调用110是被设计为从一个或多个联网设备104访问网络服务112的Web API,那么用户设备可以经由Web API调用来使用API调用110以访问服务112。类似地,API调用110可以用于访问本地API、远程过程调用或其他API。

[0032] 图2图示了根据本公开的一些方面的另一代表性系统架构200。该架构200图示了如下示例,其中,NL到API转换208不驻存在用户设备202上并且例如作为服务或其他API可到达。用户设备202包括用户接口206,如先前结合图1描述的。设备经由用户接口接收(多个)NL话语。NL话语被传递到NL到API转换208所驻存的(多个)转换系统209。NL到API转换208如本文所描述的被实现和执行。

[0033] 对于如本文所描述的实现交互性的实施例,转换系统209将信息发送到用户设备202,如本文讨论的,用户设备202被需要将各种专业化的解码器模块的输出经由用户接口呈现,并接收任何用户校正。用户校正(如果有的话)被发送回到NL到API转换208,并且NL到API转换208创建API调用210,如下面所描述的,并且将API调用210返回到如所描述的其被利用的用户设备202。

[0034] 对于在转换过程中未实现交互性的实施例,NL到API转换208将所创建的API调用210返回到其被利用的用户设备202。

[0035] 图3图示了根据本公开的一些方面的用于将自然语言输入转换成应用编程接口(API)调用的代表性流程图300。该流程图300呈现高级过程并且转换的细节在下面进行描述。

[0036] 存在被执行以将NL话语转换为输出API调用的三种主要操作。第一种操作是如由操作302图示的接收NL话语。NL话语由用户设备(诸如用户设备102或202)经由用户接口(诸如用户接口106或206)接收。NL话语可以以各种格式来接收。例如,用户设备可以以语音格式(诸如经由麦克风)接收NL话语。另外或备选地,NL话语可以以文本格式来接收。所接收的NL话语可以被转换为标准格式(诸如文本)以用于由NL到API转换过程进行处理。例如,以语音格式接收的NL话语可以通过语音到文本转换过程被转换为文本。许多语音到文本转换过程是已知的,并且任何这样的已知过程可以被用作到文本格式的转换的部分。

[0037] 另外或备选地,所接收的NL话语可以被预处理以消除错误的来源。例如,拼写可以被校正,停滞词(例如,没有语义含义但是是语法正确的NL话语的部分的词)可以被添加或移除,和/或其他这样的预处理。任何这样的预处理的目的是将(多个)话语置为转换准备要发生的格式。

[0038] 用户可以使用各种NL话语来表达相同的意图。意图是用户试图通过NL话语实现的内容。例如,假设用户的意图是找出与项目Alpha有关的所有电子邮件,并且想要它们被显示使得具有最老时间戳的电子邮件被首先显示。图3:304图示了可以由用户利用以表达这种意图的样本NL话语:

[0039] •“向我示出与项目Alpha有关的未读电子邮件,早期的在前面;”

[0040] •“未读的项目Alpha电子邮件按时间排序反转;”

[0041] •“找出我尚未阅读的与项目Alpha有关的电子邮件,最老的在前面。”

[0042] NL话语首先通过机器学习模型被映射到API框架。如下面所讨论的,本公开的实施例可以利用序列到序列模型。API框架指定API中的主要项(名称、参数、值等)而没有实际进行API调用可能需要的任何特定API格式化或其他信息,诸如协议信息。在转换中使用的API框架从NL话语被映射到的一组API被导出。例如,Web API调用可以看起来像是:

[0043] 获得(GET)https://service.domain.com/v1.0/<user-id>/messages?$filter＝isRead%20eq%20FALSE&$search＝“Top%20Project”$orderby＝receivedDateTime%20asc

[0044] 这样的API调用包括进行实际Web API调用需要的特定格式化、协议相关信息、转换等。API框架指定API中的主要项(名称、参数、值等)而没有实际进行API调用可能需要的任何特定API格式化或其他信息,诸如协议信息。例如,针对以上GET API调用的API框架可以像是:

[0045] (GET-Messages{

[0046] Filter(isRead=FALSE),

[0047] Search("Top Project"),

[0048] OrderBy(receivedDateTime,asc)})

[0049] 获得-消息{

[0050] 过滤(是否阅读=否),

[0051] 搜索("顶部项目"),

[0052] 排序(接收的日期时间,升序)}

[0053] 在将NL话语302映射到API框架306中使用的一组API框架通过首先考虑将通过NL到API转换过程而被识别的一组API(一组识别的API)来导出。一组API框架可以从一组识别的API中的每个API导出。该组API框架然后表示序列到序列模型的可能输出,因为其将NL话语302转换成API框架306。

[0054] 该组API框架还确定序列到序列模型从NL话语中识别的参数类型。例如,文章"Roy T.Fielding and Richard N.Taylor,Architectural styles and the design of network-based software architectures,University of California,Irvine Doctoral dissertation,2000"描述了用于API的REST架构风格。遵守REST架构风格的API具有表1中规定的参数类型。

[0055] 表1:API参数类型

[0056]

参数类型	描述
搜索(字符串)	搜索包含特定关键词的资源
过滤(布尔表达式)	通过由布尔表达式指定的准则进行过滤，例如，是否阅读=否
排序(属性，顺序)	按升序（'asc'）或降序（'desc'）根据属性对资源进行排序
选择(属性)	代替完整资源，返回通过指定属性选择的资源
计数()	对匹配的资源的数量进行计数
获得顶部(整数)	返回由整数值指定的资源的第一数量

[0057] 其他API架构可以得出不同组的参数类型。

[0058] 该组API框架中的参数类型影响在转换器中使用的序列到序列模型的解码器如何被因子分解(factored)成模块。这在下面更详细地进行解释。总之,解码器可以被因子分解为使得每个模块识别一种参数类型和/或一个API。

[0059] 在图3的示例中,NL话语304被映射到API框架308中。如以上所描述的并且在下面更详细地描述的,模块的输出可以形成用于校正将NL话语304转换成API框架308中的任何错误的用户交互性的基础。这经由用户接口310来完成,如在下面更详细地描述的。

[0060] 一旦NL话语304通过NL到API转换过程306被映射到API框架308中，API框架308就被映射到API 314中。这通过一组确定性映射规则312来完成。例如，解析器可以获取API框架308的不同片(piece)并且利用一组确定性映射规则将API框架308的每片映射到API调用的对应片。该过程在图7中更详细地图示。

[0061] 图4图示了根据本公开的一些方面的用于使用机器学习模型将自然语言输入转换成API调用的代表性流程图400。流程图400可以实现例如图3中图示的过程。

[0062] NL话语402由如本文所图示的系统接收。操作406表示发生在NL话语上的任何预处理，诸如从一种格式到另一格式(例如，语音到文本)的转换、对停滞词的添加和/或移除、拼写校正、词干提取和/或准备用于到API框架的初始转换的NL话语所期望的任何其他类型的预处理。

[0063] 经训练的机器学习算法被用于将NL话语转换成API框架。机器学习问题可以被表征为：给定输入话语$x=\{x_1,x_2,...,x_m\}$，其中x_i是输入话语中的第i项，将x映射到对应的线性化API框架$y=\{y_1,y_2,...,y_n\}$，其中，y_j是API框架的第j方面。这种类型的问题可以通过各种经训练的机器学习模型来解决。一个合适的模型是序列到序列神经模型。对于输入序列$x=\{x_1,x_2,...,x_m\}$，序列到序列神经模型估计针对所有可能输出序列$y=\{y_1,y_2,...,y_n\}$的条件概率分布$p(y|x)$。长度m和n可以是不同的，并且它们中的两者都可以改变。图4的编码器408和解码器410表示序列到序列神经模型的说明性示例。

[0064] 编码器408被实现为双向递归神经网络(RNN)，首先将x编码成状态向量序列(h_1, h_2,...,h_m)。如果Φ是将每个词嵌入到低维向量中的随机初始化的词嵌入层，那么前向RNN和后向RNN的状态向量分别被计算为：

[0065] $$\vec{h}_i = GRU_{fw}(\phi(x_i), \vec{h}_{i-1}) \quad (1)$$

[0066] $$\overleftarrow{h}_i = GRU_{bw}(\phi(x_i), \overleftarrow{h}_{i+1}) \quad (2)$$

[0067] 其中门控递归单元(GRU)是隐藏激活函数。合适的函数被定义在文章“Kyunghyun Cho,Bart van Merrienboer,Calar Gulcehre,Dzmitry Bahdanau,Fethi Bougares,Holger Schwenk,and Yoshua Bengio,Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translations,Proceedigns of Conference on Empirical Methods in Natural Language Processing,1724-1734,2014”中。该参考文献定义了类似于在LSTM机器学习模型中使用的函数但是可以更易于计算和实现的一种类型的隐藏激活函数。该隐藏激活函数包括重置门，使得先前隐藏状态可以被忽略。当计算第j个隐藏单元的激活时，重置门r_j通过下式来计算：

[0068] $r_j=\sigma([Wrx]_j+[U_rh_{t-1}]_j)$ (3)

[0069] 其中，σ是逻辑sigmoid函数，并且$[\cdot]_j$表示向量的第j个元素。x和h_{t-1}分别是输入和先前隐藏状态。W_r和U_r是被学习的权重矩阵。

[0070] 更新门z_j通过下式来计算：

[0071] $z_j=\sigma([W_zx]_j+[U_zh_{t-1}]_j)$ (4)

[0072] 其中加以必要的修改，元素是如上所定义的。所提议的单元h_j的实际激活通过下式来计算：

[0073] $$h_j^t = z_j h_j^{t-1} + (1 - z_j)\tilde{h}_j^t \quad (5)$$

[0074] 其中

[0075] $$\tilde{h}_j^t = \phi([\boldsymbol{W}\boldsymbol{x}]_j + [\boldsymbol{U}(\boldsymbol{r} \odot \boldsymbol{h}_{t-1}]_j) \quad (6)$$

[0076] 在这个公式中，当重置门接近0时，隐藏状态被强制忽视先前隐藏状态并仅用当前输入重置。这有效地允许隐藏状态在未来丢弃稍后被发现不相关的任何信息，因此允许更紧凑的表示。

[0077] 另一方面，更新门控制来自先前隐藏状态的多少信息将继续到当前隐藏状态。其作用类似于LSTM神经网络中的存储器并帮助RNN记住长期信息。

[0078] 由于每个隐藏单元具有单独的重置门和更新门时，每个隐藏单元将学习捕获在不同时间尺度上的依赖性。学习捕获短期依赖性的那些单元将具有最活跃的更新门。

[0079] RNN的前向和后向状态向量被联接，

[0080] $$[\vec{h}_i, \overleftarrow{h}_i], i = 1, \ldots, m.$$

[0081] 当解码器410被实现为没有因子分解的解码器模块的统一解码器时，解码器是注意力RNN，其将生成输出标记(token)，一次一个。解码器的状态向量被表示为(d_1,d_2,...,d_n)。注意力采取附加注意力的形式。对于解码步骤j，解码器被定义为：

[0082] $$d_0 = \tanh(W_0[\vec{h}_m, \overleftarrow{h}_1]) \quad (7)$$

[0083] $u_{ji} = v^T \tanh(W_1 h_i + W_2 d_j)$ (8)

[0084] $$\alpha_{ji} = \frac{u_{ji}}{\sum_{i'=1}^{m} u_{ji'}} \quad (9)$$

[0085] $$h'_j = \sum_{i=1}^{m} \alpha_{ji} h_i \quad (10)$$

[0086] $d_{j+1} = GRU([\Phi(y_j), h'_j], d_j)$ (11)

[0087] $$p(y_i | \boldsymbol{x}, y_{1:j-1}) \propto e^{(v[a_j, h'_j])} \quad (12)$$

[0088] 其中W_0、W_1、W_2、v和U是模型参数。解码器首先计算在解码器状态上的归一化注意力权重α_{ji}并获得汇总状态h'_j。汇总状态然后用于计算下一个解码器状态d_{j+1}和输出概率分布$p(y_i | x, y_{1:j-1})$。在训练期间，序列$y_{1:j-1}$使用“黄金”输出序列来提供。在测试期间，其由解码器生成。

[0089] 解码器的结果可以在用户界面(UI)412中被呈现给用户以允许针对API框架的校正、修改和/或接受的用户交互性。统一解码器架构(诸如以上已经描述的)不允许因子分解的解码器架构允许的细粒度的交互性。这是因为统一解码器的每个预测y_j基于涉及整个输入序列x和所有先前输出标记$y_{1:j-1}$的复杂计算。

[0090] 用于因子分解的解码器架构的UI在下面被呈现。对于统一解码器架构，UI 412允许用户验证结果(例如，整个API框架)。作为备选方案，在下面更详细地描述的用于因子分解的解码器架构的相同UI可以被呈现并且用户可以修改、编辑或批准API框架的各个方面。

[0091] 一旦任何用户输入经由UI 412接收，或者如果因为没有交互性被使用和/或期望，没有信息被呈现给用户，那么API框架到API调用映射414使用如下面描述的确定性映射规则来执行。结果是包括做出相关联的API调用所需要的所有必要格式化和其他信息的API调

用416。

[0092] 图5图示了根据本公开的一些方面的用于使用机器学习模型将自然语言输入转换成API框架的代表性流程图500。在该架构中，解码器520被因子分解成解码器模块，其中的每个被训练以识别预定义种类的输入序列。

[0093] 在图5的架构中，输入502由编码器504接收。编码器504以与以上已经描述的图4的编码器408相同的方式被实现并操作。编码器504被实现为双向递归神经网络(RNN)，首先将x编码成状态向量序列(h_1,h_2,...,h_m)。

[0094] 为了实现与API框架的更细粒度的交互，解码器520包括多个解码器模块506、508、510、512。在预测预定义种类的输出方面，每个模块是专业化的(例如，被训练)。换言之，通过读取如由编码器504编码的输入NL话语，每个模块被训练以识别API框架的特定部分。因为模块508、510、512与API框架的特定部分相关联，所以用户可以容易地理解每个模块的预测并可以在模块级与系统交互。

[0095] 解码器中的模块是被设计为履行特定序列预测任务的专业化的神经网络。当将解码器因子分解成模块时，因子分解基于应当是针对解码器520的一组可能输出框架的一组API框架发生。例如，考虑被设计为与家中的灯和其他物联网(IOT)设备交互的一组API框架。这些设备可以例如使用集线器经由语音控制、文本控制等被控制。假设一组API框架是：

[0096] 1.(GET-Status{

[0097] device(deviceID),

[0098] location(Location)})

[0099] 获得-状态{

[0100] 设备(设备ID),

[0101] 位置(位置)}

[0102] 2.(SET-Status{

[0103] device(deviceID),

[0104] status(isON)})

[0105] 设置-状态{

[0106] 设备(设备ID),

[0107] 状态(是否开启)}

[0108] 3.(SET-Status{

[0109] device(deviceID),

[0110] status(isONLINE)})

[0111] 设置-状态{

[0112] 设备(设备ID),

[0113] 状态(是否在线)

[0114] 这些API框架可以被设置为对应于一组因子分解的解码器，诸如：

[0115] (a)API-获得-状态：其生成获得-状态API框架的名称；

[0116] (a)STATUS设备ID：其生成获得-状态API框架的设备ID参数；

[0117] (b)STATUS位置：其生成获得-状态API框架的位置参数；

[0118] (d)API-设置-状态开启：其生成具有是否开启参数的设置-状态API框架的名称；

[0119] (e)STATUS开启ID:其生成具有是否开启参数的设置-状态API框架的设备ID参数;

[0120] (f)STATUS开启:其生成具有是否开启参数的设置-状态API框架的是否开启参数;

[0121] (g)API-设置-状态在线:其生成具有是否在线参数的设置-状态API框架的名称;

[0122] (h)STATUS在线ID:其生成具有是否在线参数的设置-状态API框架的设备ID参数;

[0123] (i)STATUS在线:其生成具有是否在线参数的设置-状态API框架的是否在线参数。

[0124] 这些不是一组API框架可以被因子分解的唯一方式。目标是将解码器因子分解成与一组目标API框架中的参数和各种API相对应的一组模块,使得API框架部分的个体概念(名称、参数等)可容易理解,如下面更详细地描述的。

[0125] 因此,因子分解的解码器中的不同模块对应于API框架的不同部分,诸如一组目标API框架中的每个API框架的名称、参数等。作为另一示例,对于具有以下可能参数的代表性获得-消息API框架:按发送者过滤(过滤(发送者))、通过已读或未读过滤(过滤(是否阅读))、选择消息是否具有相关联的附件(选择(附件))、按日期和时间排序(排序(接收的日期时间))、以及搜索具有关键词的消息(搜索(关键词)),解码器将对API框架进行因子分解,因此存在识别API的API模块、针对发送者的过滤模块、针对是否阅读的过滤模块、排序模块、以及搜索模块。当参数是可选的时,可选参数可以在解码期间被激活或可以不被激活。因此,总体上,解码器被因子分解成与识别API框架集中的API的一个或多个模块和针对API框架集中的每个API中的每个参数的模块相对应的模块。

[0126] 在被触发时,模块的任务是读取如由编码器504编码的输入话语并实例化(识别)完整参数。为了这么做,模块需要基于编码的输入话语来确定其参数值。例如,给定输入话语“与博士研究有关的未读电子邮件”,搜索模块需要预测搜索参数的值是“博士研究”,并生成完整参数“搜索博士研究”作为输出序列。类似地,过滤(是否阅读)模块需要学习如“未读电子邮件”、“尚未被阅读的电子邮件”以及“还没有阅读的电子邮件”的短语全部指示其参数值是否。

[0127] 因为解码器中的每个模块都具有清楚定义的语义,所以在模块级实现用户交互变得简单直接。形式上,模块M_k是如公式(7)-(12)中定义的注意力解码器,其目标是估计条件概率分布$p(y|x)$,其中,y来自一组API框架符号。

[0128] 对于任何给定NL话语,仅仅几个模块将被触发。控制器模块506确定哪些模块要触发。具体地,控制器还被实现为注意力解码器。使用经编码的NL话语作为输入,其生成被称为布局(layout)的模块的序列。模块然后被激活并且生成它们相应的参数,并且最终参数被组成以形成最终API框架。形式上,控制器是如公式(7)-(12)中定义的注意力解码器,其目标是估计条件概率分布$p_o(l|x)$,其中,布局l来自一组模块。

[0129] 在图5的代表性示例中,控制器读取经编码的输入话语并生成模块的序列:API 508、过滤(是否阅读)510、和搜索512。每个模块被激活并读取经编码的输入话语以生成它们相应的参数,其中主要工作是基于话语来确定正确参数值。因此,API 508生成获得-消息514,过滤(是否阅读)510生成过滤是否阅读为否516,并且搜索512生成搜索项目Alpha 518。

[0130] API 508、过滤(是否阅读)510、和搜索512模块可以被容易地映射到用户可以容易地理解的解释。API模块508生成将被调用的API的名称。过滤(是否阅读)模块510选择被阅读/未被阅读的消息。搜索模块512搜索具有相关联的关键词的消息。

[0131] 对于因子分解的解码器520，对于由$\{(\boldsymbol{x}_i, \boldsymbol{l}_i, \boldsymbol{y}_i)\}_{i=1}^{N}$给出的特定组训练示例(训练数据)，损失函数Θ包括三种损失，其一起由下式给出：

[0132] $$\Theta = \frac{1}{N}\sum_{i=1}^{N}(\Theta_{c,i} + \Theta_{m,i}) + \lambda\Theta_{L2} \qquad (13)$$

[0133] 对于第i个示例，控制器损失是由下式给出的布局预测上的交叉熵损失：

[0134] $\Theta_{c,i} = -\log p_c(l_i|x_i)$ (14)

[0135] 假设第i个示例相对于参数$\{y_{i,1}, y_{i,2}, \ldots, y_{i,t}\}$的“黄金”布局(基本事实布局)$l_i = \{M_1, M_2, \ldots, M_t\}$，模块损失是模块预测上的平均交叉熵损失：

[0136] $$\Theta_{m,i} = -\frac{1}{t}\sum_{j=1}^{t}\log p_j(\boldsymbol{y}_{i,j}|\boldsymbol{x}_i) \qquad (15)$$

[0137] 最后，L2正则化项Θ_{L2}以平衡参数λ被添加以减轻过拟合。如文章“Geoffrey E.Hinton,Nitish Srivastava,Alex Krizhevsky,Ilya Sutskever,and Ruslan R.Salkhutdinov,Improving Neural Networks by Preventing Co-adaptation of Feature Detectors,arXiv:1207.0580[cs.NE]2012”中讨论的丢弃(Dropout)也可以被应用。如该文章中所讨论的，丢弃通过随着训练数据点被提交而随机地移除神经网络中的隐藏单元来应用。在每个训练例(case)的每个演示上，每个隐藏单元以0.5的概率从网络被随机地省略，因此隐藏单元无法依赖于存在的其他隐藏单元。这防止隐藏单元之间的协同适应并帮助减轻过拟合。

[0138] 图6图示了根据本公开的一些方面的用于使用机器学习模型将自然语言输入转换成API框架的代表性流程图600。该流程图非常类似于图5中的流程图，并且以上的原理和讨论中的许多也适用于该图。该流程图中的主要差别在于因子分解的解码器618不具有控制器。

[0139] 在图6的实施例中，编码器以与如图5中描述的相同方式接收NL话语602并编码话语，并且以上的相同讨论同样在该实施例中适用。

[0140] 解码器618如以上所讨论的被因子分解有解码器模块，其中每个解码器模块表示API框架集(其表示解码器的输出)中的API框架的属性。每个因子分解的解码器模块如以上所讨论的被训练。然而，不存在控制器模块。因为不存在控制器模块来标识布局并激活布局中的解码器模块，所以每个个体解码器模块在适当的输入序列被识别时自激活。因此，输入序列(编码器的输出)被发送到整个集合中的每个解码器模块。这些模块接收输入，并且当适当的输入序列被识别时，它们自激活并产生适当的输出，如以上所描述的。因此，在图6的示例(其是图5的相同示例)中，API模块606识别API的名称并生成获取-消息612，过滤(是否阅读)608生成过滤是否阅读为否614，并且搜索610生成搜索项目Alpha 616。

[0141] 被激活的模块的输出被聚合为最终输出API框架。另外，解码器模块可以被映射到交互式UI，如本文所讨论的。

[0142] 一旦API框架由机器学习模型生成，如以上所描述的，API框架就被映射到API调用。图7图示了用于将API框架映射到API调用的代表性流程图700。该映射通过确定性规则来完成，其将API框架的部分映射到API调用中的对应部分。在图7的实施例中，解析器用于解析出API框架的各个部分，并且然后规则用于将所解析的部分映射到API调用的部分。这些部分然后被汇编以产生API调用。

[0143] 作为代表性示例,图7示出了以下API框架:

[0144] (GET-Messages{

[0145] Filter(isRead=FALSE),

[0146] Search("Project Alpha"),

[0147] OrderBy(receivedDateTime,asc)})

[0148] 获得-消息{

[0149] 过滤(是否阅读=否),

[0150] 搜索("项目Alpha"),

[0151] 排序(接收的日期时间,升序)}

[0152] 到以下API调用的映射:

[0153] 获得

[0154] https://service.domain.com/v1.0/<user-id>/messages?$filter=isRead%20eq%20FALSE&$search="Project%20Alpha"&$orderby=receivedDateTime%20asc

[0155] 解析器704被适配以将API框架解析成它们的组成部分。因此,对于输入API框架702,第一部分是API 406的名称。这被解析为"获得-消息"706。'{'字符指示已经到达名称的结尾并且参数将在接下来到来。

[0156] 一组确定性规则将API框架的不同名称映射到API调用的名称。因此,API名称706被映射到API调用名称708以及将API框架转变成可调用API需要的格式化和其他信息。因此,在代表性示例中,API框架名称"获得-消息"708被映射到API调用名称"获得https://service.domain.com/v1.0/<user-id>/messages?"710。为了完成映射,该组确定性规则可以利用附加信息。例如,在以上API调用中,参数<user-id>(<用户-id>)不在API框架中并且可以从用户简档或其他源提取。这些附加数据源可以提供不在API框架中并且不是映射规则的部分的附加信息。

[0157] 经解析的API框架的下一部分是过滤参数类型。这在API框架中被指定为"过滤(是否阅读=否)"710。这被映射到API调用中的针对参数的对应机制。在代表性示例中,其是$filter parameter"$filter=isRead%20eq%20FALSE"712。当解析器确定附加参数仍然要到来时,接合'&'字符可以被置于输出中。

[0158] 经解析的API框架的下一部分是搜索参数类型。这由"搜索("项目Alpha")"714指定。其通过确定性规则被映射到针对对应参数的API调用,$search。因此,搜索参数类型被映射到"$search="Project%20Alpha""716。同样,一旦解析器识别到附加参数类型存在,则'&'符号可以被置于输出流中。

[0159] 最后的参数类型是"排序(接收到的日期时间,升序)"718,其被映射到对应的API调用参数类型"$orderby=receivedDateTime%20asc"720。'}'符号向解析器指示已经找出API框架的结尾并且解析器可以然后将API调用输出流的各个部分汇编成对应的API调用722。

[0160] 在以上示例中,经解析的API框架的每个部分具有API调用中的对应部分。在一些情形下,API调用可以需要在API框架中未指定的显式参数。例如,API框架可以标识可选地指定的参数并且假定针对参数(其对于可选参数不是显式的)的默认值。然而,实际API调用需要针对所有可选参数的值。在这种情形下,规则可以添加所需要的参数以及默认值以确

保API调用完整并且符合必要的调用惯例、格式化、参数要求等。

[0161] 这在API框架具有参数并且该相同参数可选地在API调用中具有默认值的情况下也适用。在这种情形下，如果API框架中的参数的值与API调用中的默认值匹配，那么参数可以被API调用忽视。

[0162] 图8图示了示出根据本公开的一些方面的用于使用机器学习模型将自然语言输入转换成API调用的交互式用户界面806的代表性示图800。利用因子分解的解码器，个体解码器模块可以被映射到易于理解的概念以用于呈现在交互式用户界面806上。例如，使用获得-消息API框架的示例：

[0163] 获得-消息{

[0164] 过滤(是否阅读＝否)，

[0165] 搜索(“项目Alpha”)，

[0166] 排序(接收的日期时间,升序)}

[0167] 可能已经产生API框架的个体模型是生成名称(获得-消息)的API模块、生成拉取已读/未读电子邮件(是否阅读＝否)的过滤器的过滤(是否阅读)模块、生成搜索字符串(“项目Alpha”)的搜索模块、以及生成按升序的接收的日期/时间对所接收的邮件进行排序的参数的排序模块。由这些模块中的每个模块生成的输出被映射到对用户来说容易理解的描述。

[0168] 交互式用户界面806可以包括用于用户输入NL话语的输入字段808或其他机制，诸如激活麦克风以接收说出话语的按钮。一旦话语完成，用户就可以指示完成，诸如通过按压控件(例如，“搜索”控件)、通过等待一段时间、或者话语完成的任何其他指示。

[0169] 话语被发送到如以上所讨论的转换过程，并且个体解码器模块中的每个产生它们的输出。UI 810的交互式部分基于哪些模块被激活(例如，通过布局或通过自激活)而被汇编。因为每个模块具有针对用户的对应的解释，所以一旦知道哪些模块被触发以及它们相关联的输出，UI 810的交互式部分的个体字段和控件就可以被汇编。因此，UI 810的交互式部分以非常直接的方式表示因子分解的解码器的因子分解的输出，并且可以在不同调用之间改变以使得所显示的与由机器学习模型产生的一样容易理解。

[0170] 在以上获得-消息API框架中，存在被显示在交互式UI 810中的若干代表性映射。获得-消息名称被分解成两个字段。“获得”向用户容易地解释为如图中示出的“找出”。“消息”部分可以向用户解释为“消息”或“电子邮件”，或将允许用户快速地看到作为NL话语被输入的与机器学习模型如何识别NL话语的该方面之间的相关性的描述符。

[0171] 过滤(是否)输出以及所生成的参数“否”容易地由交互式UI 810的“是未读的”方面来解释。类似地，搜索(“项目Alpha”)的搜索模块输出被容易地解释为如图示的“包含关键词:项目,Alpha”。

[0172] 交互式UI 810可以包含控件，该控件允许用户调整所生成的输出应当是什么，以便将机器学习模型的所生成的输出与用户输入NL话语时所想要的进行对准。由于交互式UI 810的细粒度控制，所以用户可以在细粒度的基础上校正所生成的输出。

[0173] 控件可以包括例如移除被激活的模块的控件812。因此，如果在布局或自激活中存在错误并且本不应当被激活的模块被激活，那么用户可以使用移除控件812来移除模块。

[0174] 控件可以还包括添加本应当被激活但是没有被激活的模块的控件。因此，如果布

局或自激活遗漏了本应当被生成的输出的模块，那么本应当由遗漏的模块生成的输出可以经由添加控件816被添加。

[0175] 除了添加和/或移除以外，模块的输出可以在它们本应当被激活并且被激活了但是产生了不正确的输出的情况下被编辑。这例如通过编辑控件814来示出。

[0176] 另外或备选地，一些模块可以被训练以生成多个输出参数。作为代表性示例，搜索模块802可以具有可能来自输入NL话语的若干候选。例如，输入到808中的NL话语可以使搜索模块802生成“搜索项目alpha”、“搜索项目”和“搜索alpha电子邮件”作为针对输出参数的可能选项。输出选项可以具有也由模块生成的相关联的排序或分数。这些选项804可以被呈现在交互式UI 810中，诸如在下拉框中，使得最可能的选项被显示。如果用户对所显示的选项不满意，那么用户可以点击下拉控件(在交互式UI中的与每个字段相关联的向下箭头)并选择另一选项。对于具有有限数量的固定选项的参数，诸如可以为是或否的布尔参数，该组固定选项可以被显示在下拉框中，以使得用户可以选取不是用户想要的所生成的选项的不同值。

[0177] 一旦用户做出了任何期望的改变，用户就可以使用提交控件818或他们不进行(进一步)改变的另一指示符来指示他们对API框架满意。

[0178] 由于从解码器模块到交互式UI的细粒度的映射，API框架可以被容易地更新，并且然后API框架可以被映射到API调用，如先前所讨论的。

[0179] 尽管交互式UI 810被图示为以文本格式呈现，但是其他格式也可以被使用，诸如声音(语音)或格式的组合，诸如语音和文本的组合。

[0180] 利用因子分解的解码器来执行测试，并且结果表明仅使用少量交互，交互式UI极大地改进预测准确性。利用仅一轮交互，测试准确性从大约0.5改进到超过0.9。另外，在由用户执行的比较研究中，交互式UI与非交互式的对应物(counterpart)相比较，其中“黑盒”解码器相对交互式UI(诸如以上所描述的，其呈现因子分解的输出并允许用户进行改变)，产生未被因子分解的输出。在黑盒解码器中，如果输出是错误的，那么用户没有选择而只能重新制定NL话语并再次尝试。在因子分解的输出中，如果它是错误的，那么交互式UI允许用户做出调整并提交经调整的查询。测试数据示出，交互式UI导致更高的任务成功率、更短的任务完成时间(更少的用户努力)以及显著更高的用户满意率。大多数参与者表示他们相比于重新制定NL话语并再次尝试更喜欢上述交互式UI。

[0181] 在这些测试中，因子分解的解码器使用在“Yu Su,Ahmed Hassan Awadallah,Madian Khabasa,Patrick Pantel,Michael Gamon,and Mark Encarnacion,Building Natural Language Interfaces to Web APIs,Proceedings of the International Conference on Information and Knowledge Management,2017”中发布的公共可用数据集NL2API被训练。在训练模型中，数据被拆分成训练集和测试集。另外，20％的数据被保留以用作验证集。

[0182] 在第一测试中，其中NL话语被输入并且模型的输出与本应当被输出的进行比较，结果发现，利用单次调整(例如，经由UI添加模块、经由UI移除模块、经由UI调整参数)，准确性从大约0.5增加到超过0.9，附加交互进一步增加准确性。另外，在与非交互式、非因子分解的解码器和与因子分解的解码器相关联的交互式UI的人类交互中，用户极其喜欢交互式UI和因子分解的解码器。与仅35％的用户报告他们对非交互式模式满意或强烈满意相比，

60％的用户对交互式模式满意或强烈满意。与非交互式模式相比，对于交互式模式，感知到的额外努力好得多。与70％的使用交互式模式的参与者报告他们需要很少额外努力来完成任务相比，仅仅25％的使用非交互式模式的参与者报告他们需要很少额外努力来完成任务。另外，对于交互式UI，实现正确API框架采取的动作的数量和实现正确框架的时间小得多。下面的表2呈现了结果。在该表中，不成功的尝试是其中用户在完成之前放弃了任务并移动到另一任务。

[0183] 表2：完成或放弃的动作和时间

[0184]

UI	成功	#动作	完成或放弃的动作和时间
非交互式	否	6.39	119.08
非交互式	是	4.67	84.08
非交互式	全部	5.10	92.83
交互式	否	4.30	47.40
交互式	是	3.45	29.81
交互式	全部	3.73	35.54

[0185] 收集训练数据以训练在NL话语到API转换过程中的机器学习模型可以是有点劳动密集的。如下面所描述的，对序列到序列神经模型的训练使用监督式训练过程。监督式训练使用经注释的数据，其对应于输入数据和对应的期望输出。对于具有控制器的实施例，经注释的数据还包括适当的布局。收集这样的经注释的数据可以例如利用一组注释者和/或众包工作者。他们将获取各种NL话语，并且在具有一组API框架的知识的情况下，标识NL话语对应于哪个API框架以及参数、针对那些参数的值，以及对于具有控制器的实施例，适当的布局。然而，考虑到可以映射到相同API框架和相同参数和值的各种各样的NL话语，确保充分的覆盖使得各种各样的不同的NL话语被覆盖可能有点困难。这样可以是相对劳动密集的且困难的。交互式UI和细粒度控制提供了极好机制，以收集正在进行的训练数据以允许冷启动类型部署情景或在部署经初始训练的机器学习模型之后的进一步调整。

[0186] 图9图示了900来自根据本公开的一些方面的用于使用机器学习模型将自然语言输入转换成API调用的代表性交互式用户界面904的反馈908。该高级示图示出了附加训练数据可以如何被收集。

[0187] 如以上所指出的，交互式用户界面904包括直接与因子分解的解码器(诸如解码器520和618)中的经激活的解码器模块的输出相对应的不同的字段、值等。由于细粒度的控制和用户具有的反馈以及对个体解码器模块的直接对应关系，所以一旦用户已经针对API框架908做出了任何校正并提交了最终信息，所提交的数据就可以用作附加训练数据点。有效地，用户成为针对所提交的NL话语的注释者，示出机器学习模型本应当如何生成API框架。另外，因为所提交的校正直接对应于各种解码器模块的输出，所以所提交的数据908示出了本应当由经激活的解码器模块中的每个产生的输出和正确布局。

[0188] 此所提交的信息908可以由NL到API转换过程902和/或另一系统和/或过程聚合使

得机器学习模型可以被更新以减少未来转换906中的错误。

[0189] 图10图示了根据本公开的一些方面的利用机器学习模型训练的代表性系统架构1000。该架构1000示出了交互式UI可以如何用于收集训练数据用于API转换的代表性实施例。

[0190] 如所讨论的，NL话语由用户设备1002的用户接口1006接收。NL话语被传递到NL到API转换过程1008。如以上所讨论的，一旦NL话语被解码成API框架。因子分解的解码器模块的输出经由用户接口1006在交互式UI中被呈现给用户。如先前所描述的，用户做出任何调整并将经调整的API框架提交到NL到API转换过程1008。

[0191] 经调整的API框架使用如先前所描述的确定性过程和/或规则被映射以产生API调用1010。

[0192] 一旦经调整的API框架已经被提交，NL到API转换过程1008就具有经注释的数据点，该注释的数据点包括输入NL话语、正确解码的API框架、以及针对每个解码器模块的正确输出、和/或正确布局。该数据点可以被保存并聚合在本地并被发送到编码器/解码器训练更新机器1014直到触发事件为止，或在它们被收集时被传递到编码器/解码器训练更新机器1014。触发事件可以是以下中的一个或多个：时间的推移(诸如周期性或非周期性排程)、收集的数据点的数量、使用的本地存储量、发送所收集的数据点的规划带宽、用户设备1002与编码器/解码器训练更新机器1014之间的数据连接的类型，等。

[0193] 在编码器/解码器训练更新机器1014上接收的数据可以由训练/更新过程1016聚合，并且在触发事件后，所聚合的数据可以用于更新机器学习模型、与机器学习模型相关联的参数或两者。数据可以基于用户群来聚合或跨所有用户聚合。用户群是具有一个或多个共同方面(诸如在用户简档、地理相邻性、人口统计相似性、和/或任何其他类型的相似性的一个或多个参数之中的共性)的用户的组。触发事件可以是以下各项中的一项或多项：时间的推移，诸如周期性或非周期性排程、收集的数据点的数量、使用的本地存储量，等。

[0194] 训练/更新过程1016将经更新的参数、经更新的模型、或两者分发给合适的用户。合适的用户是针对其训练/更新了模型的那些用户。一旦经更新的参数和/或经更新的模型被接收到，NL到API转换过程1008就利用它们用于从NL话语到API框架的转换。

[0195] 图11图示了根据本公开的一些方面的用于机器学习模型训练的代表性训练过程1100。机器学习是在计算机不被显式编程的情况下给予计算机学习的能力的研究领域。机器学习探索可以从现有数据中学习并做出关于新数据的预测的算法(在本文中也被称为工具)的研究和构建。这样的机器学习工具通过根据示例训练数据1108建立模型1112来操作，以便做出被表示为输出或评价(如API框架1120)的数据驱动的预测或决策。尽管示例实施例使用序列到序列神经模型被呈现，但是其他实施例可以利用不同的机器学习模型。

[0196] 在一些示例实施例中，不同的机器学习工具可以被使用。例如，逻辑回归(LR)、朴素贝叶斯、随机森林(RF)、神经网络(NN)、矩阵因子分解、以及支持向量机(SVM)工具可以用于产生API框架。

[0197] 总体上，在机器学习中存在两种类型的问题：分类问题和回归问题。分类问题旨在将项分类到若干类别中的一种(例如，该物体是苹果还是橘子?)。回归问题旨在量化一些项(例如，通过提供为实数的值)。在一些实施例中，示例机器学习算法提供从NL话语到API框架的映射。机器学习算法利用训练数据1108来找出影响从NL话语到API框架的映射的所标

识的特征1102之间的相关性。

[0198] 在一个示例中，特征1102可以是不同类型的并且可以包括API名称1104、(多个)API参数、以及如本文描述的其他特征。

[0199] 利用训练数据1108和所标识的特征1102，机器学习模型1112在操作1110处被训练。机器学习模型评估特征1102的值，因为它们与训练数据1108相关。训练的结果是经训练的机器学习模型1112。

[0200] 在经训练的机器学习模型1112被产生之后，验证操作可以在一些实例中被执行。验证涉及采取一组经注释的数据1114并且使用经训练的机器学习模型1112来产生针对验证数据集1114中的数据点中的每个数据点的输出。针对该集合的输出与描述了本应当从机器学习模型1112输出的内容的注释进行比较。统计数据可以被评价以看出经训练的机器学习模型1112操作地有多好，并且如果准确性可接受，那么经训练的机器学习模型1112可以被验证1116。否则，附加训练可以被执行。

[0201] 当机器学习模型1112或1116用于执行评定(assessment)时，新数据1118被提供为对经训练的机器学习模型1116的输入，并且机器学习模型1116生成诸如API框架1120的评定作为输出。

[0202] 对于本公开的实施例，监督式学习过程被使用。初始训练使用可用数据来完成，可用数据诸如先前描述的NL2API数据集。随着附加数据诸如通过本文讨论的交互式UI被收集，附加的经注释的数据可以用于进一步训练。这个的优点在于当用户的输入NL话语诸如通过词汇偏移等而偏移时，机器学习模型可以被更新以说明这些变化。

[0203] 另外，随着新的API被添加到一组API或现有API被改变，具有新的和/或经更新的因子分解的解码器模块的新机器学习模型可以被分发。另外，因为经注释的数据经由交互式UI被产生，所以新的解码器模块(例如，用于处理新的或经更新的API)可以以冷启动方式被部署而无需训练和/或需要很少训练，并且随着附加数据被收集，解码器模块训练被更新。

[0204] 示例机器架构和机器可读介质

[0205] 图12图示了适合于实现本文公开的系统和其他方面或适合于执行本文公开的方法的代表性机器架构。图12的机器被示出为独立设备(诸如本文描述的移动设备)，其适合于实施以上的构思。对于以上描述的服务器方面，在数据中心中运行的多个这样的机器、云架构的部分等可以被使用。在服务器方面中，不是所有的所说明的功能和设备都被利用。例如，尽管用户用来与服务器和/或云架构交互的系统、设备等可以具有屏幕、触摸屏输入等，但是服务器常常不必具有屏幕、触摸屏、照相机等，并且通常通过具有合适输入和输出方面的连接系统与用户交互。因此，下面的架构应当被理解为包含多种类型的设备和机器，并且各种方面可以存在于或可以不存在于任何特定设备或机器中，取决于其形状因子和目的(例如，服务器很少具有相机，而可穿戴装置很少包括磁盘)。然而，图12的示例解释适合于允许本领域技术人员确定在对所图示的实施例的使用的特定设备、机器等进行适当修改的情况下，如何利用硬件和软件的适当组合来实现先前描述的实施例。

[0206] 尽管仅单个机器被图示，但是术语“机器”还应被理解为包括单独地或联合地执行一组(或多组)指令以执行本文讨论的方法中的一种或多种方法的机器的任何汇集。

[0207] 机器1200的示例包括至少一个处理器1202(例如，中央处理单元(CPU)、图形处理

单元(GPU)、高级处理单元(APU)或其组合)、一个或多个存储器(诸如主存储器1204、静态存储器1206、或其他类型的存储器),其经由链路1208与彼此通信。链路1208可以是总线或其他类型的连接通道。机器1200可以还包括可选方面,诸如包括任何类型的显示器的图形显示单元1210。机器1200可以还包括其他可选方面,诸如字母数字输入设备1212(例如,键盘、触摸屏等)、用户界面(UI)导航设备1214(例如,鼠标、轨迹球、触摸设备等)、存储单元1216(例如,磁盘驱动器或(多个)其他存储设备)、信号生成设备1218(例如,扬声器)、(多个)传感器1221(例如,全球定位传感器、(多个)加速度计、(多个)麦克风、(多个)照相机、眼动跟踪子系统等)、输出控制器1228(例如,与一个或多个其他设备连接和/或通信的有线或无线连接(诸如通用串行总线(USB)、近场通信(NFC)、红外(IR)、串行/并行总线等))、以及用于(例如,有线和/或无线)连接到一个或多个网络1226和/或通过一个或多个网络1226进行通信的网络接口设备1220。

[0208] 不是更常规的微处理器,神经网络芯片可以用于实现本公开的实施例。神经网络芯片是被设计用于执行各种形式的神经网络并且可以被使用在序列到序列神经模型或在实施例中利用的其他机器学习模型中的专用芯片。因此,它们适合用于实现本公开的方面(诸如本公开的机器学习模型和其他神经网络方面)。基于本文包含的公开内容,本领域技术人员将知道如何使用一个或多个神经网络芯片来实现本公开的实施例。

[0209] 可执行指令和机器存储介质

[0210] 各种存储器(例如,1204、1206、和/或(多个)处理器1202的存储器)和/或存储单元1216可以存储体现或由本文描述的方法或功能中的任何一个或多个所利用的一个或多个指令集和数据结构(例如,软件)1224。这些指令在由(多个)处理器1202执行时使各种操作实现所公开的实施例。

[0211] 如本文所使用的,术语“机器存储介质”、“设备存储介质”、“计算机存储介质”意指同样的事物并且可以在本公开中可互换使用。这些术语指代存储可执行指令和/或数据的单个或多个存储设备和/或介质(例如,集中式或分布式数据库、和/或相关联的高速缓存和服务器)。这些术语因此应被理解为包括存储设备,诸如固态存储器,以及光学和磁性介质,包括处理器内部或外部的存储器。机器存储介质、计算机存储介质和/或设备存储介质的具体示例包括非易失性存储器,其通过示例半导体存储器器件的方式包括:例如,电可编程只读存储器(EPROM)、电可擦可编程只读存储器(EEPROM))、FPGA以及闪存器件;诸如内部硬盘和可移除盘的磁盘;磁光盘;以及CD-ROM和DVD-ROM盘。术语机器存储介质、计算机存储介质、以及设备存储介质具体地且明确地包括载波、经调制的数据信号、以及其他这样的瞬态介质,其中的至少一些被涵盖在下面讨论的术语“信号介质”下。

[0212] 信号介质

[0213] 术语“信号介质”应当被理解为包括任何形式的经调制的数据信号、载波等。术语“经调制的数据信号”意指设置或改变其特性中的一个或多个使得在信号中编码信息的信号。

[0214] 计算机可读介质

[0215] 术语“机器可读介质”、“计算机可读介质”和“设备可读介质”意指同样的事物并且可以在本公开中可互换使用。这些术语被定义为包括机器存储介质和信号介质两者。因此,术语包括存储设备/介质和载波/经调制的数据信号两者。

[0216] 示例实施例

[0217] 示例1.一种计算机实现的方法,包括:

[0218] 接收自然语言话语;

[0219] 将自然语言话语提交到包括序列到序列神经模型的经训练的机器学习模型,序列到序列神经模型包括编码器和多个解码器,多个解码器中的每个解码器被耦合到编码器并且被训练为识别从编码器输出的一个或多个标记,并且将一个或多个标记映射到API框架的一个或多个项;

[0220] 从经训练的机器学习模型接收多个项,每个项从不同的编码器被接收;

[0221] 将多个项汇编到API框架中,API框架表示自然语言话语与最终API格式之间的中间格式;

[0222] 将API框架映射到最终API格式;以及

[0223] 使用最终API格式来发出对API的调用。

[0224] 示例2.根据示例1的方法,其中编码器包括递归神经网络。

[0225] 示例3.根据示例1的方法,其中每个解码器包括注意力递归神经网络。

[0226] 示例4.根据示例1的方法,还包括被耦合到编码器的控制器,控制器接收编码器的输出并产生包括多个解码器的布局并激活布局中的解码器中的每个解码器。

[0227] 示例5.根据示例1的方法,还包括:

[0228] 经由用户界面呈现每个项,用户界面包括:

[0229] 用于呈现每个项以及指示项意指什么的相关联的指示的区域;

[0230] 控件,被激活时移除相关联的项;

[0231] 控件,被激活时添加新的项;

[0232] 经由用户界面接收指示对多个项的任何改变完成的输入。

[0233] 示例6.根据示例5的方法,还包括:

[0234] 聚合执行数据,执行数据包括:

[0235] 自然语言话语;

[0236] 经汇编的API框架;以及

[0237] 经由用户界面对多个项做出的任何改变。

[0238] 示例7.根据示例1的方法,其中将API框架映射到最终API格式使用将API框架的多个项确定性地映射到最终API格式中的多个项的多个规则来完成。

[0239] 示例8.根据示例1的方法,其中编码器和多个解码器使用监督式学习过程被训练,监督式学习过程利用经注释的自然语言话语数据,经注释的自然语言话语数据包括:

[0240] 多个训练自然语言话语;以及

[0241] 针对训练自然语言话语中的每个训练自然语言话语的API框架。

[0242] 示例9.根据示例1的方法,还包括使用利用交互式用户界面收集的数据来更新对序列到序列神经模型的训练,交互式用户界面呈现从多个解码器输出的多个项并允许:

[0243] 项在将多个项汇编到API框架中之前被移除;

[0244] 项在将多个项汇编到API框架中之前被添加;以及

[0245] 项在将多个项汇编到API框架中之前被修改。

[0246] 示例10.一种系统,包括:

[0247] 处理器和具有可执行指令的设备存储介质,可执行指令在由处理器运行时使系统执行包括以下的操作:

[0248] 接收自然语言话语;

[0249] 将自然语言话语提交到包括序列到序列神经模型的经训练的机器学习模型,序列到序列神经模型包括编码器和多个解码器,多个解码器中的每个解码器被耦合到编码器并且被训练为识别从编码器输出的一个或多个标记,并且将一个或多个标记映射到API框架的一个或多个项;

[0250] 从经训练的机器学习模型接收多个项,每个项从不同的编码器被接收;

[0251] 将多个项汇编到API框架中,API框架表示自然语言话语与最终API格式之间的中间格式;

[0252] 将API框架映射到最终API格式;以及

[0253] 使用最终API格式来发出对API的调用。

[0254] 示例11.根据示例10的系统,其中编码器包括递归神经网络。

[0255] 示例12.根据示例10的系统,其中每个解码器包括注意力递归神经网络。

[0256] 示例13.根据示例10的系统,还包括被耦合到编码器的控制器,控制器:

[0257] 接收编码器的输出;

[0258] 产生包括多个解码器的布局;以及

[0259] 激活布局中的解码器中的每个解码器。

[0260] 示例14.根据示例10的系统,还包括:

[0261] 经由用户界面呈现每个项,用户界面包括:

[0262] 用于呈现每个项以及指示项意指什么的相关联的指示的区域;

[0263] 控件,被激活时移除相关联的项;

[0264] 控件,被激活时添加新的项;

[0265] 经由用户界面接收指示对多个项的任何改变完成的输入。

[0266] 示例15.根据示例14的系统,还包括:

[0267] 聚合执行数据,执行数据包括:

[0268] 自然语言话语;

[0269] 经汇编的API框架;以及

[0270] 经由用户界面对多个项做出的任何改变;以及

[0271] 将所聚合的数据发送到系统以被用于更新序列到序列神经模型的训练。

[0272] 示例16.一种计算机实现的方法,包括:

[0273] 接收自然语言话语;

[0274] 将自然语言话语提交到包括序列到序列神经模型的经训练的机器学习模型,序列到序列神经模型包括编码器和多个解码器,多个解码器中的每个解码器被耦合到编码器并且被训练为识别从编码器输出的一个或多个标记,并且作为响应产生API框架的项;

[0275] 从经训练的机器学习模型接收多个项,每个项从不同的编码器被接收;

[0276] 将多个项汇编到API框架中,API框架表示自然语言话语与最终API格式之间的中间格式;

[0277] 将API框架映射到最终API格式;以及

[0278] 使用最终API格式来发出对API的调用。

[0279] 示例17.根据示例16的方法，其中编码器包括递归神经网络。

[0280] 示例18.根据示例16或17的方法，其中每个解码器包括注意力递归神经网络。

[0281] 示例19.根据示例16、17或18的方法，还包括被耦合到编码器的控制器，控制器接收编码器的输出并产生包括多个解码器的布局，并激活布局中的解码器中的每个解码器。

[0282] 示例20.根据示例16、17、18或19的方法，还包括

[0283] 经由用户界面呈现每个项，用户界面包括：

[0284] 用于呈现每个项以及指示项意指什么的相关联的指示的区域；

[0285] 控件，被激活时移除相关联的项；

[0286] 控件，被激活时添加新的项；

[0287] 经由用户界面接收指示对多个项的任何改变完成的输入。

[0288] 示例21.根据示例20的方法，还包括：

[0289] 聚合执行数据，执行数据包括：

[0290] 自然语言话语；

[0291] 经汇编的API框架；以及

[0292] 经由用户界面对多个项做出的任何改变。

[0293] 示例22.根据示例16、17、18、19、20或21的方法，其中将API框架映射到最终API格式使用将API框架的多个项确定性地映射到最终API格式中的多个项的多个规则来完成。

[0294] 示例23.根据示例22的方法，还包括：从不是API框架的附加数据源抽取信息并将信息置为最终API格式。

[0295] 示例24.根据示例22的方法，还包括：使用解析器来将API框架映射到最终API格式。

[0296] 示例25.根据示例16、17、18、19、20、21、22、23或24的方法，其中编码器和多个解码器使用监督式学习过程被训练，监督式学习过程利用经注释的自然语言话语数据，经注释的自然语言话语数据包括：

[0297] 多个训练自然语言话语；以及

[0298] 针对训练自然语言话语中的每个训练自然语言话语的API框架。

[0299] 示例26.根据示例16、17、18、19、20、21、22、23、24或25的方法，还包括：使用利用交互式用户界面收集的数据来更新对序列到序列神经模型的训练，交互式用户界面呈现从多个解码器输出的多个项并允许：

[0300] 项在将多个项汇编到API框架中之前被移除；

[0301] 项在将多个项汇编到API框架中之前被添加；以及

[0302] 项在将多个项汇编到API框架中之前被修改。

[0303] 示例27.根据示例16、17、18、19、20、21、22、23、24、25或26的方法，还包括：

[0304] 接收经更新的序列到序列神经模型、与序列到序列神经模型相关联的经更新的参数、或两者；以及

[0305] 利用经更新的序列到序列神经模型、经更新的参数、或两者来更新序列到序列神经模型。

[0306] 示例28.根据示例16、17、18、19、20、21、22、23、24、25、26或27的方法，还包括：

[0307] 将解码器的子集中的每个解码器映射到用户界面中的对应区域；

[0308] 将由子集中的每个解码器产生的项呈现到对应区域；以及

[0309] 呈现与每个对应区域相关联的控件，控件在由用户激活时允许用户修改由对应解码器产生的项。

[0310] 示例29.一种装置，包括用于执行根据任何前述示例的方法的工具。

[0311] 示例30.一种机器可读存储装置，包括机器可读指令，机器可读指令在被执行时实现根据任何前述示例的方法或实现根据任何前述示例的装置。

[0312] 结论

[0313] 鉴于本发明的原理和前述示例可以被应用的许多可能实施例，应当意识到，本文中描述的示例意味着仅仅是说明性的并且不应当被理解为限制本发明的范围。因此，如本文中所描述的本发明预见到如可以落入随附权利要求及其任何等效方案的范围内的所有这样的实施例。

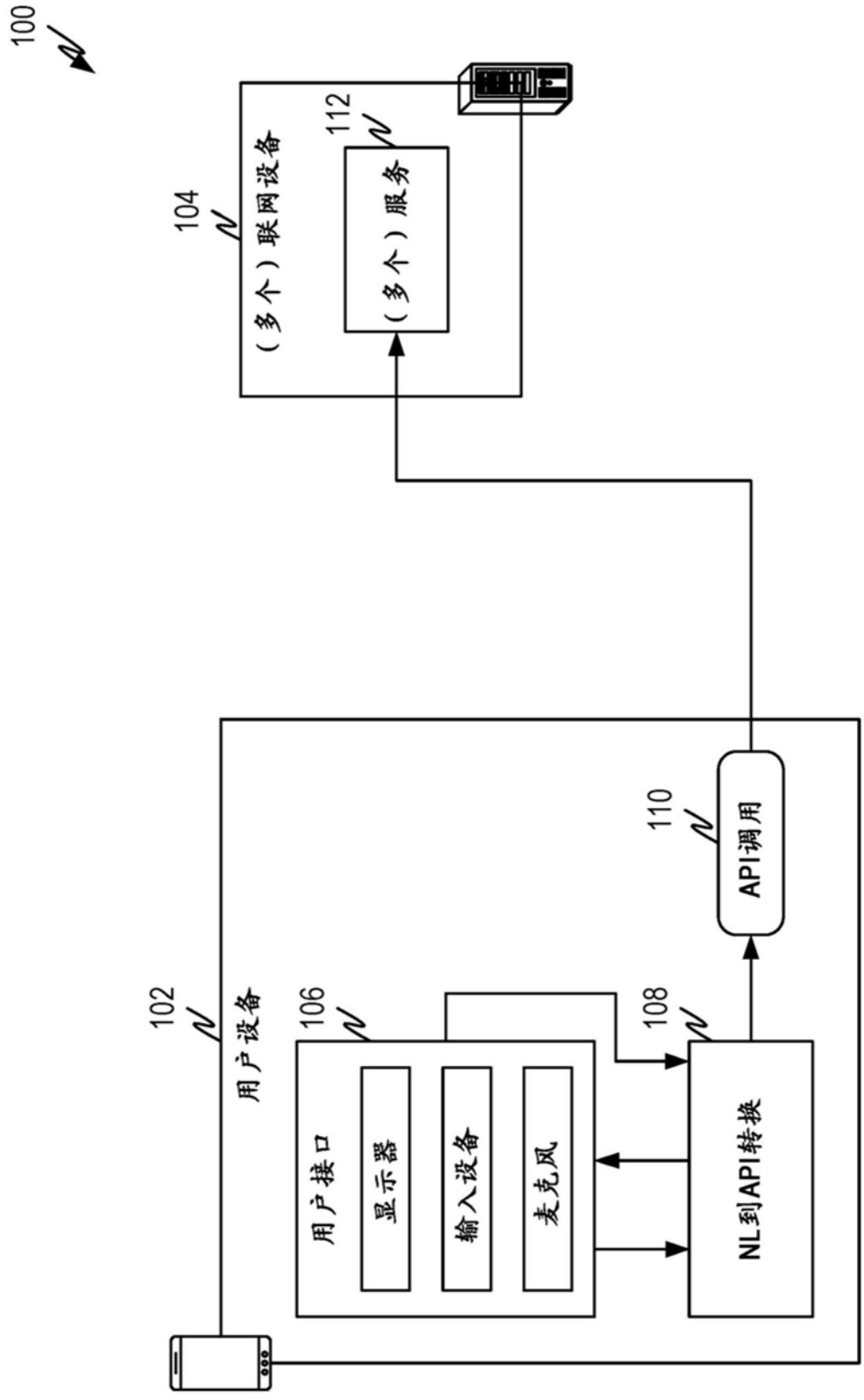

100
104
(多个)联网设备
112
(多个)服务
102
用户设备
106
用户接口
显示器
输入设备
麦克风
108
NL到API转换
110
API调用

图1

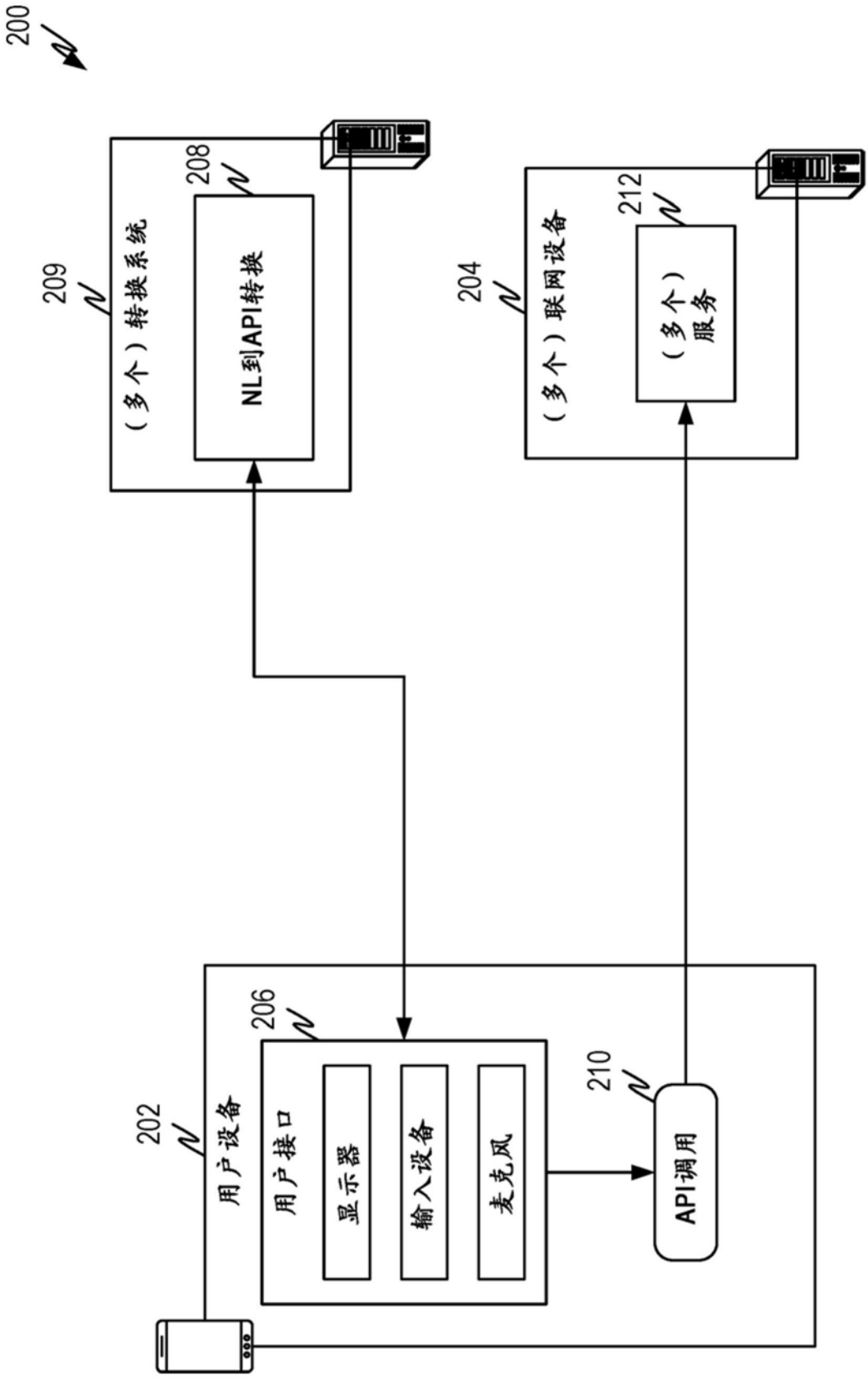

200
209
（多个）转换系统
208
NL到API转换
204
（多个）联网设备
212
（多个）服务
202
用户设备
206
用户接口
显示器
输入设备
麦克风
210
API调用

图2

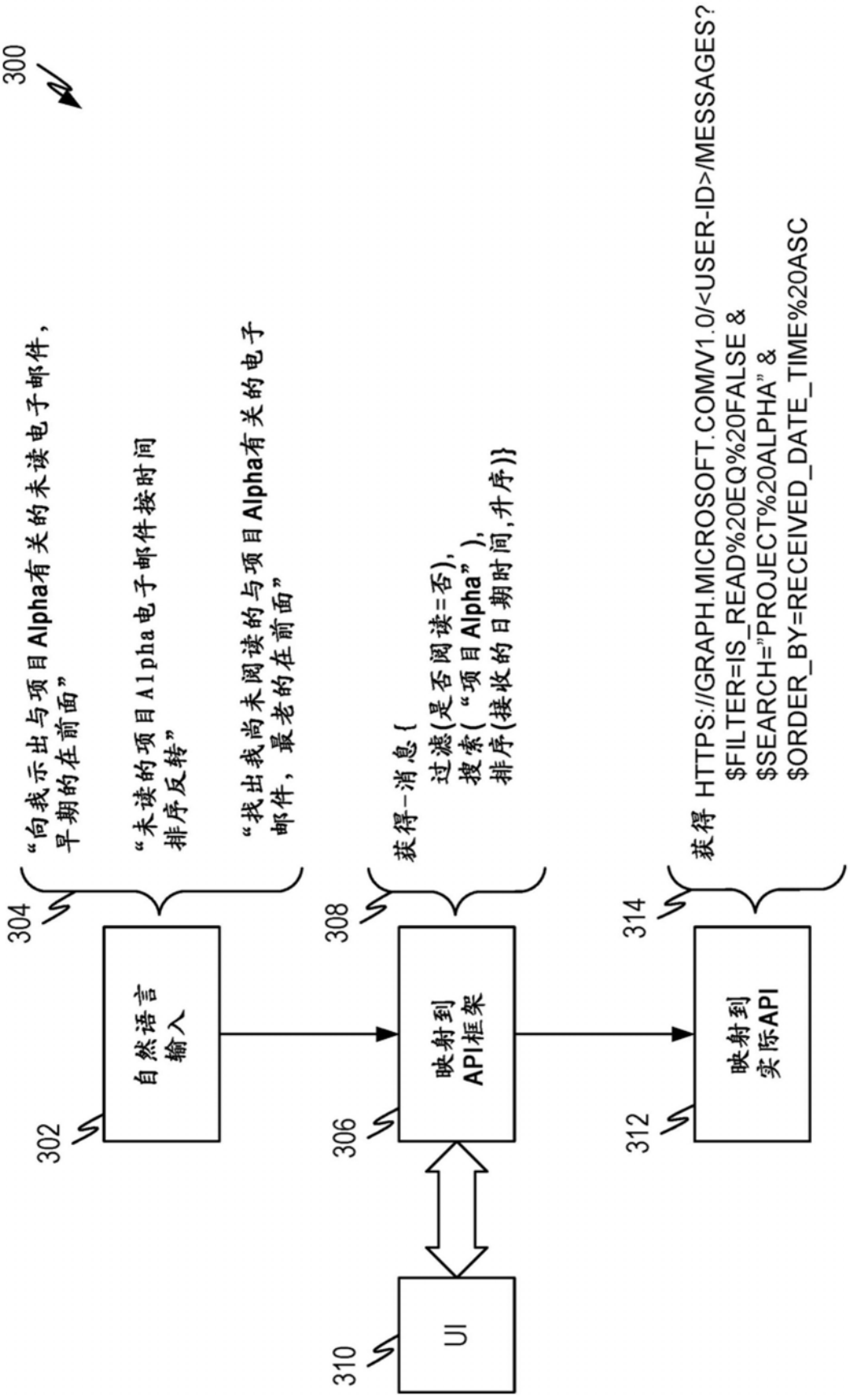

300
“向我示出与项目Alpha有关的未读电子邮件，
早期的在前面”
“未读的项目Alpha电子邮件按时间
排序反转”
“找出我尚未阅读的与项目Alpha有关的电子
邮件，最老的在前面”
304
自然语言
输入
302
获得-消息{
过滤(是否阅读=否),
搜索(“项目Alpha”),
排序(接收的日期时间,升序)}
308
映射到
API框架
306
获得 HTTPS://GRAPH.MICROSOFT.COM/V1.0/<USER-ID>/MESSAGES?
$FILTER=IS_READ%20EQ%20FALSE &
$SEARCH=”PROJECT%20ALPHA” &
$ORDER_BY=RECEIVED_DATE_TIME%20ASC
314
映射到
实际API
312
UI
310

图3

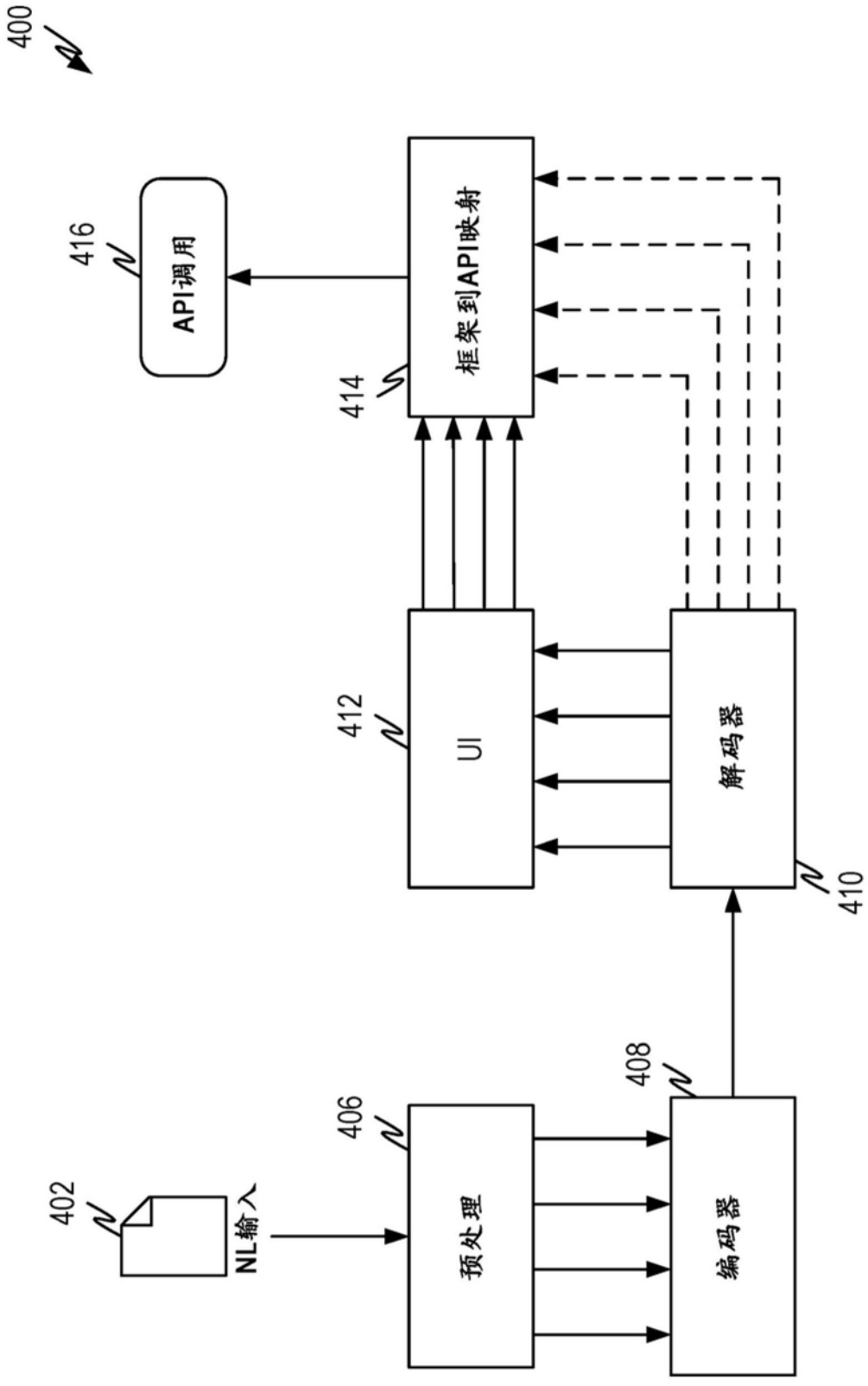

400
416
API调用
414
框架到API映射
412
UI
410
解码器
406
预处理
408
编码器
402
NL输入

图4

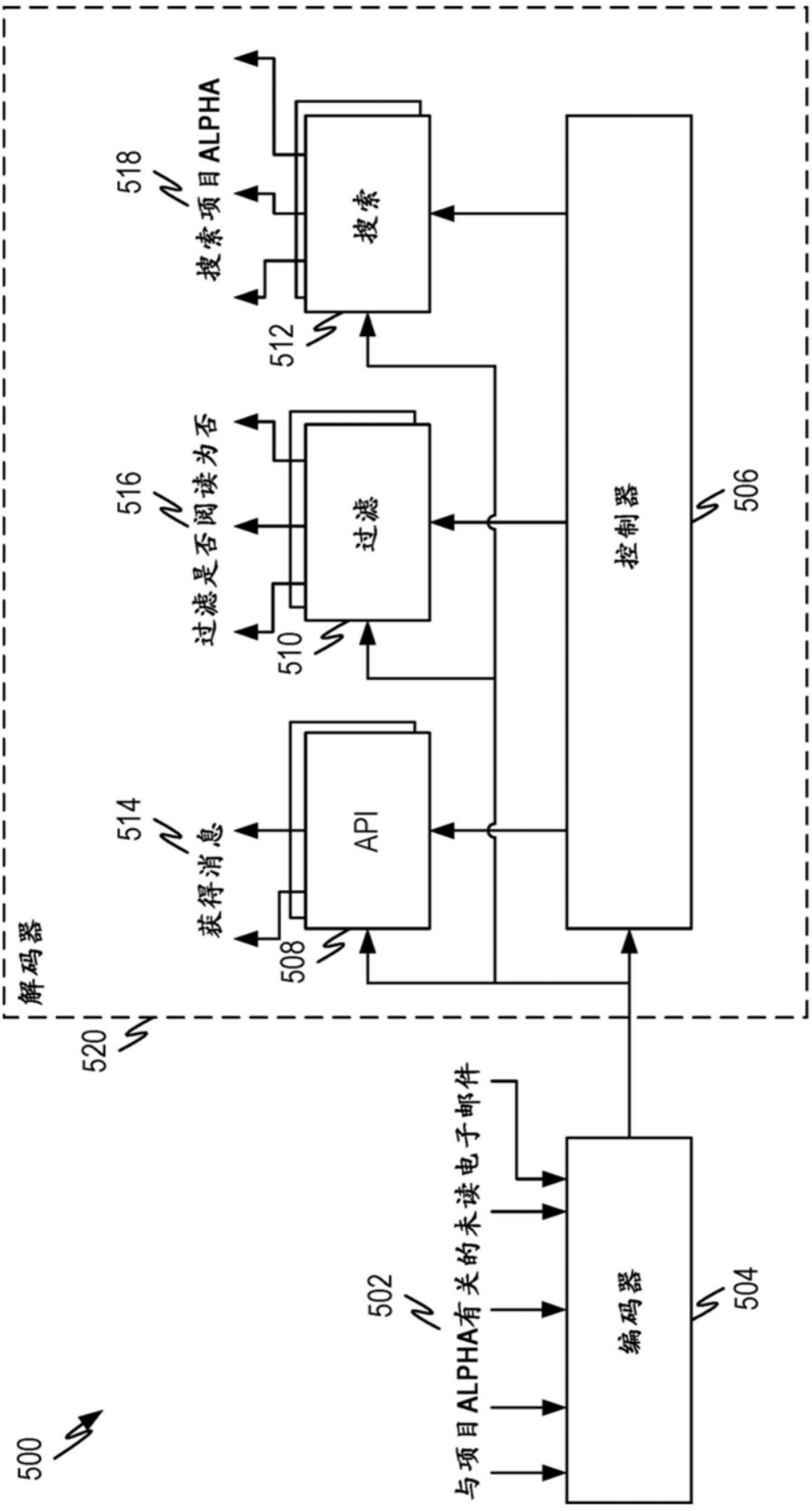

500
502
与项目ALPHA有关的未读电子邮件
编码器
504
控制器
506
解码器
520
API
508
获得消息
514
过滤
510
过滤是否阅读为否
516
搜索
512
搜索项目ALPHA
518

图5

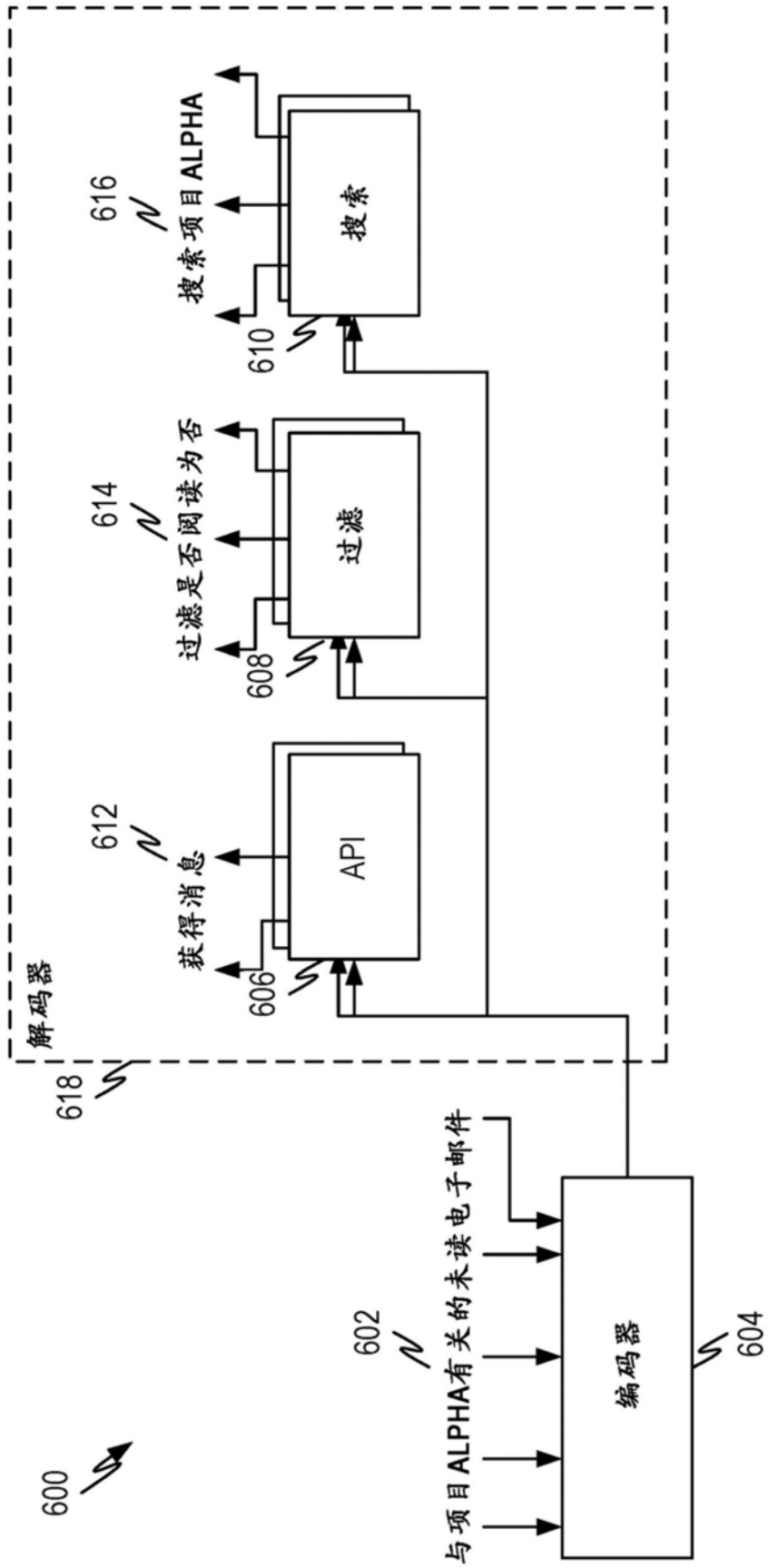

600
602
与项目ALPHA有关的未读电子邮件
编码器
604
618
解码器
606
获得消息
612
API
608
过滤是否阅读为否
614
过滤
610
搜索项目ALPHA
616
搜索

图6

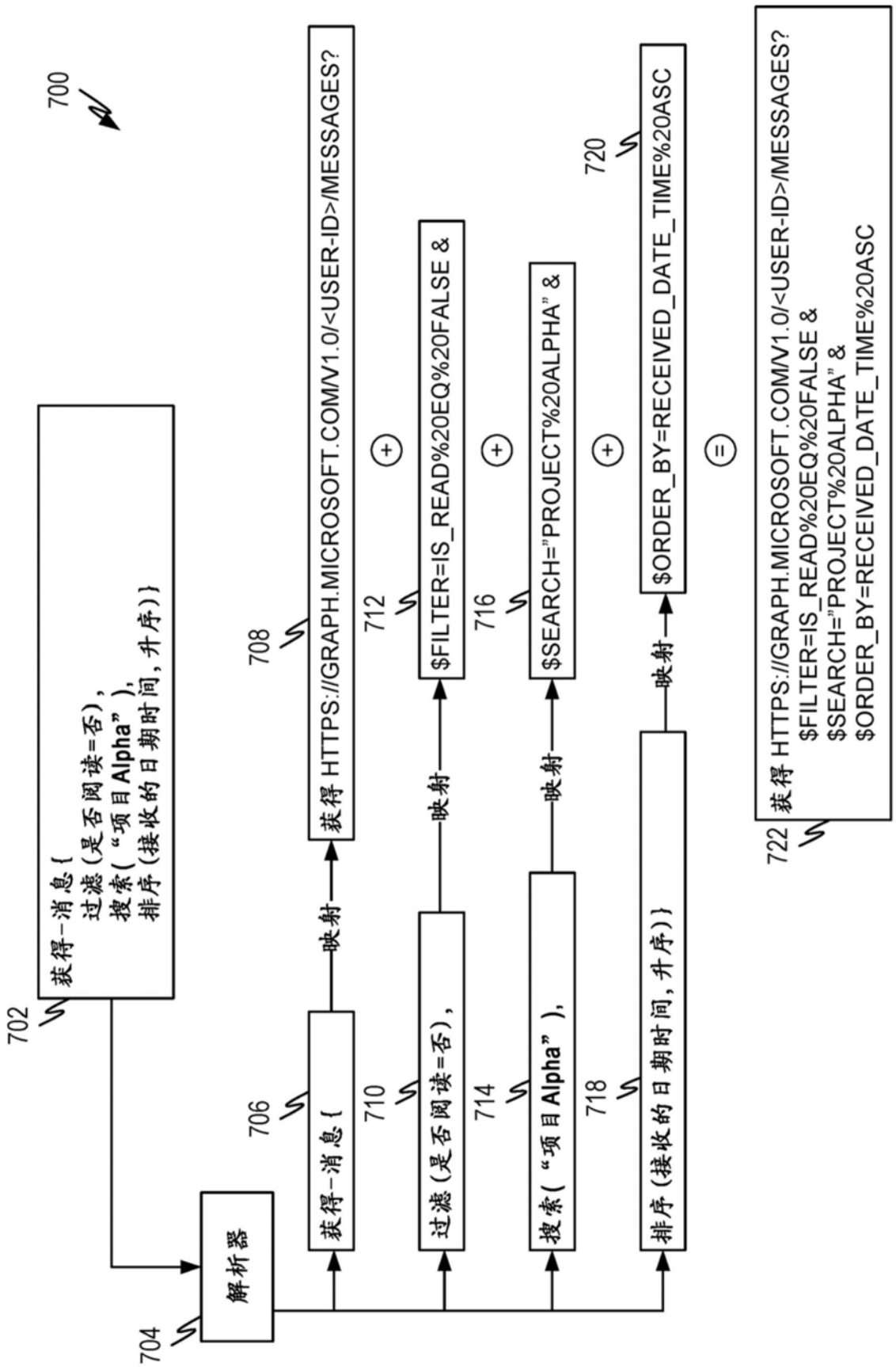

700
获得-消息{
过滤(是否阅读=否),
搜索(“项目Alpha”),
排序(接收的日期时间,升序)}
702
解析器
704
获得-消息{
706
映射
获得 HTTPS://GRAPH.MICROSOFT.COM/V1.0/<USER-ID>/MESSAGES?
708
+
过滤(是否阅读=否),
710
映射
$FILTER=IS_READ%20EQ%20FALSE &
712
+
搜索(“项目Alpha”),
714
映射
$SEARCH=”PROJECT%20ALPHA” &
716
+
排序(接收的日期时间,升序)}
718
映射
$ORDER_BY=RECEIVED_DATE_TIME%20ASC
720
=
获得 HTTPS://GRAPH.MICROSOFT.COM/V1.0/<USER-ID>/MESSAGES?
$FILTER=IS_READ%20EQ%20FALSE &
$SEARCH=”PROJECT%20ALPHA” &
$ORDER_BY=RECEIVED_DATE_TIME%20ASC
722

图7

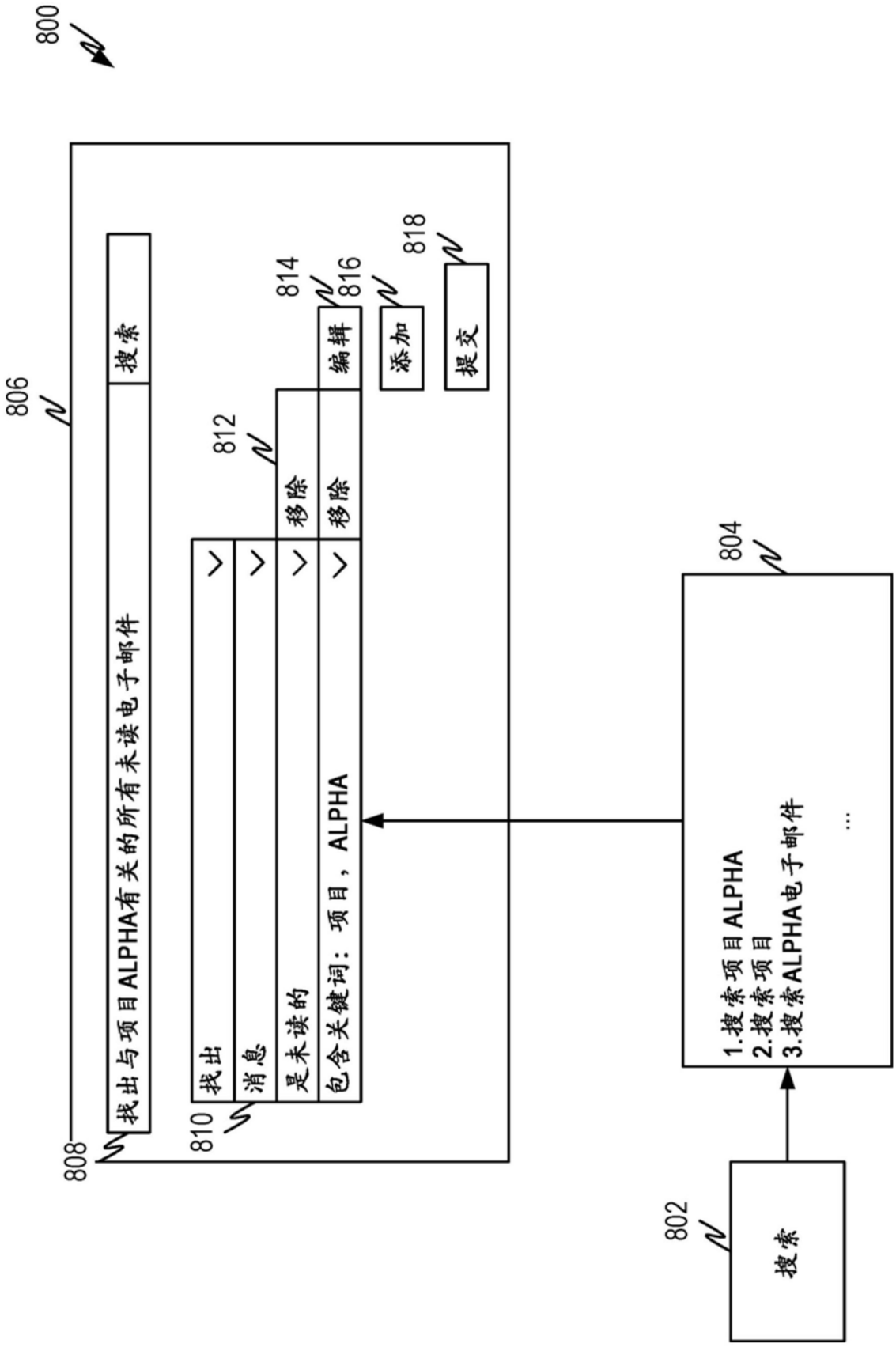

800
806
找出与项目ALPHA有关的所有未读电子邮件
搜索
808
找出
消息
是未读的
包含关键词：项目，ALPHA
810
812
移除
移除
814
编辑
816
添加
818
提交
804
1.搜索项目ALPHA
2.搜索项目
3.搜索ALPHA电子邮件
...
802
搜索

图8

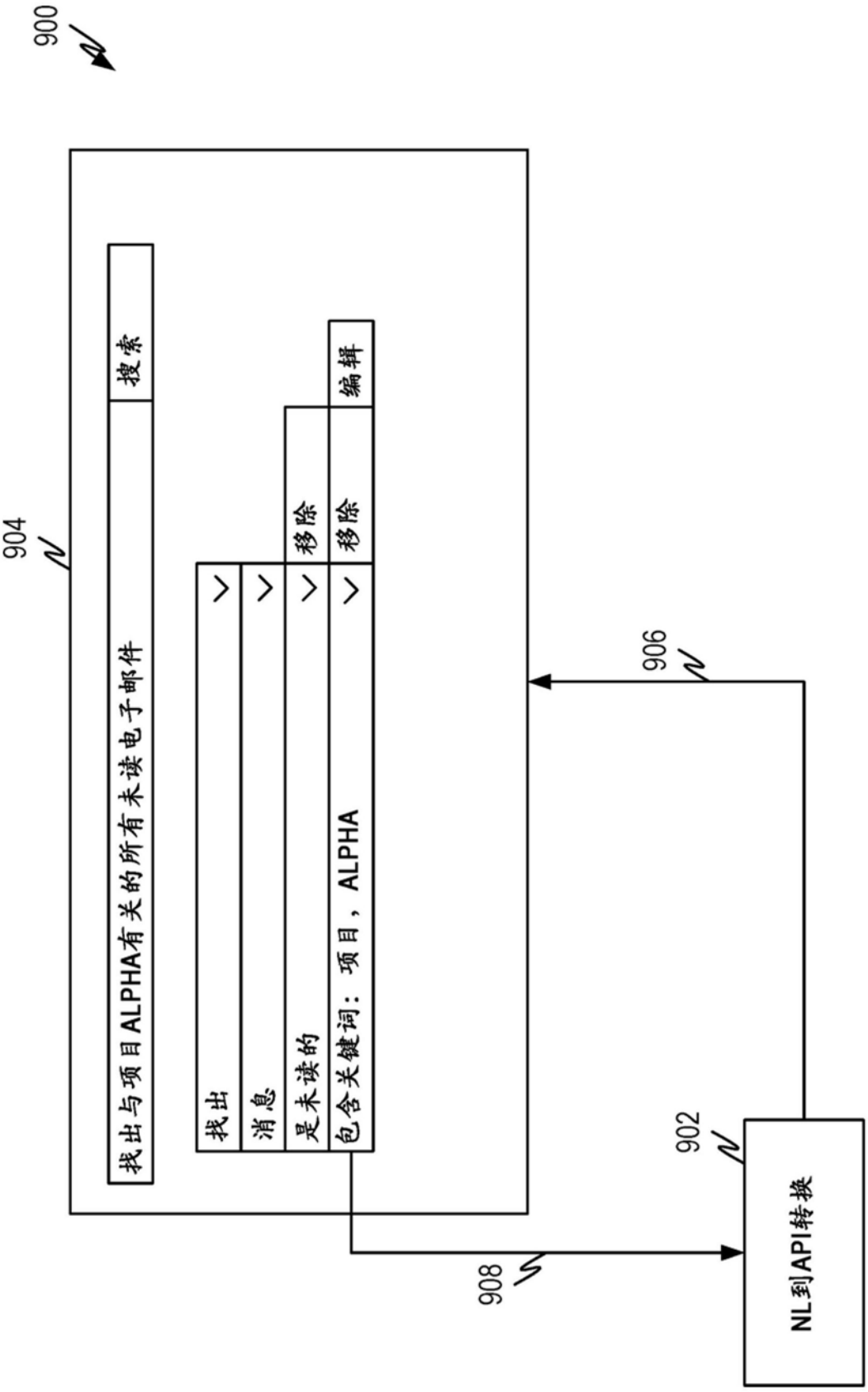

900
904
找出与项目ALPHA有关的所有未读电子邮件
搜索
找出
消息
是未读的
移除
包含关键词：项目，ALPHA
移除
编辑
906
902
NL到API转换
908

图9

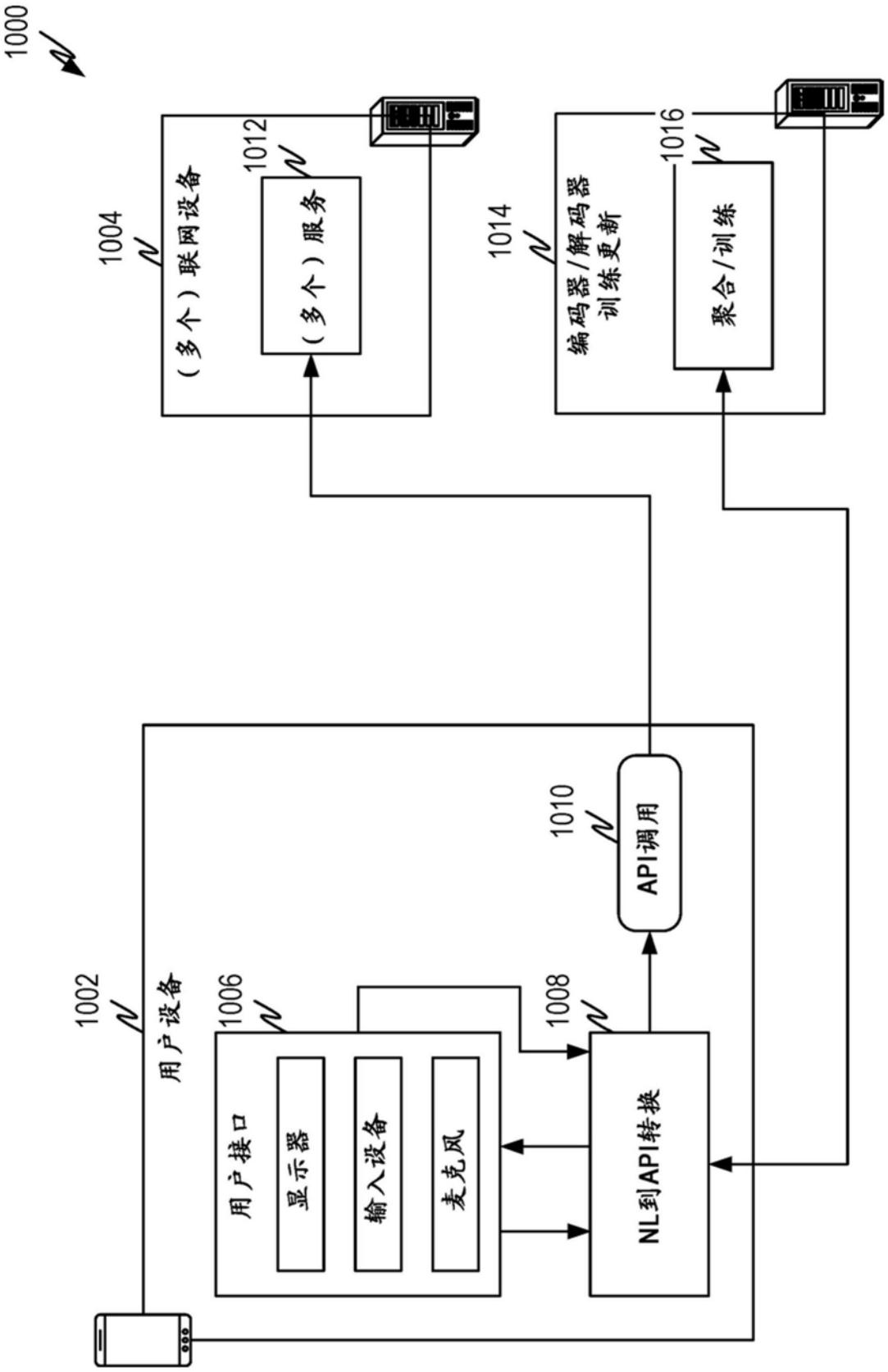

1000
1004
（多个）联网设备
1012
（多个）服务
1014
编码器/解码器
训练更新
1016
聚合/训练
1010
API调用
1002
用户设备
1006
用户接口
显示器
输入设备
麦克风
1008
NL到API转换

图10

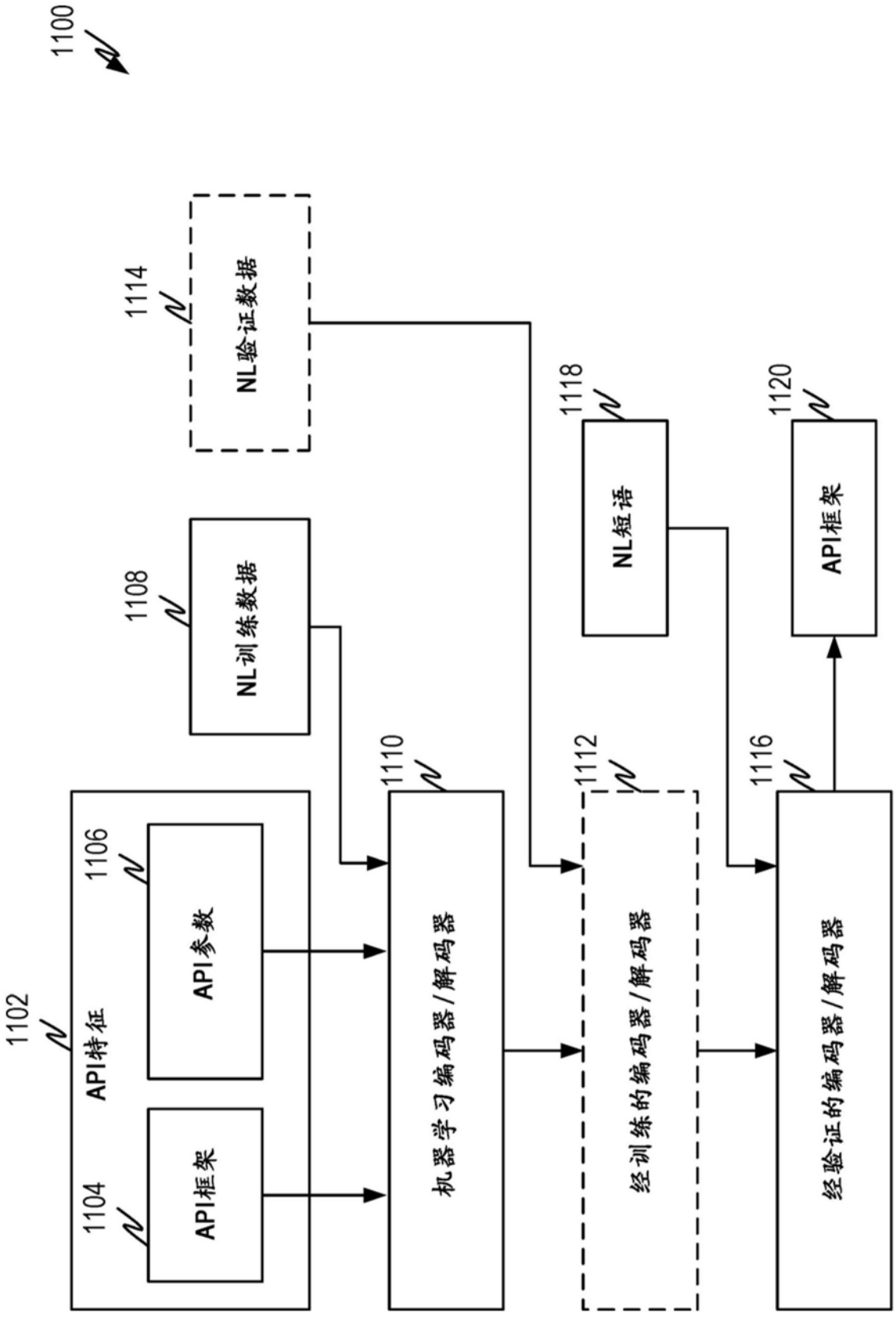

1100
1114
NL验证数据
1108
NL训练数据
1118
NL短语
1120
API框架
1102
API特征
1106
API参数
1104
API框架
1110
机器学习编码器/解码器
1112
经训练的编码器/解码器
1116
经验证的编码器/解码器

图11

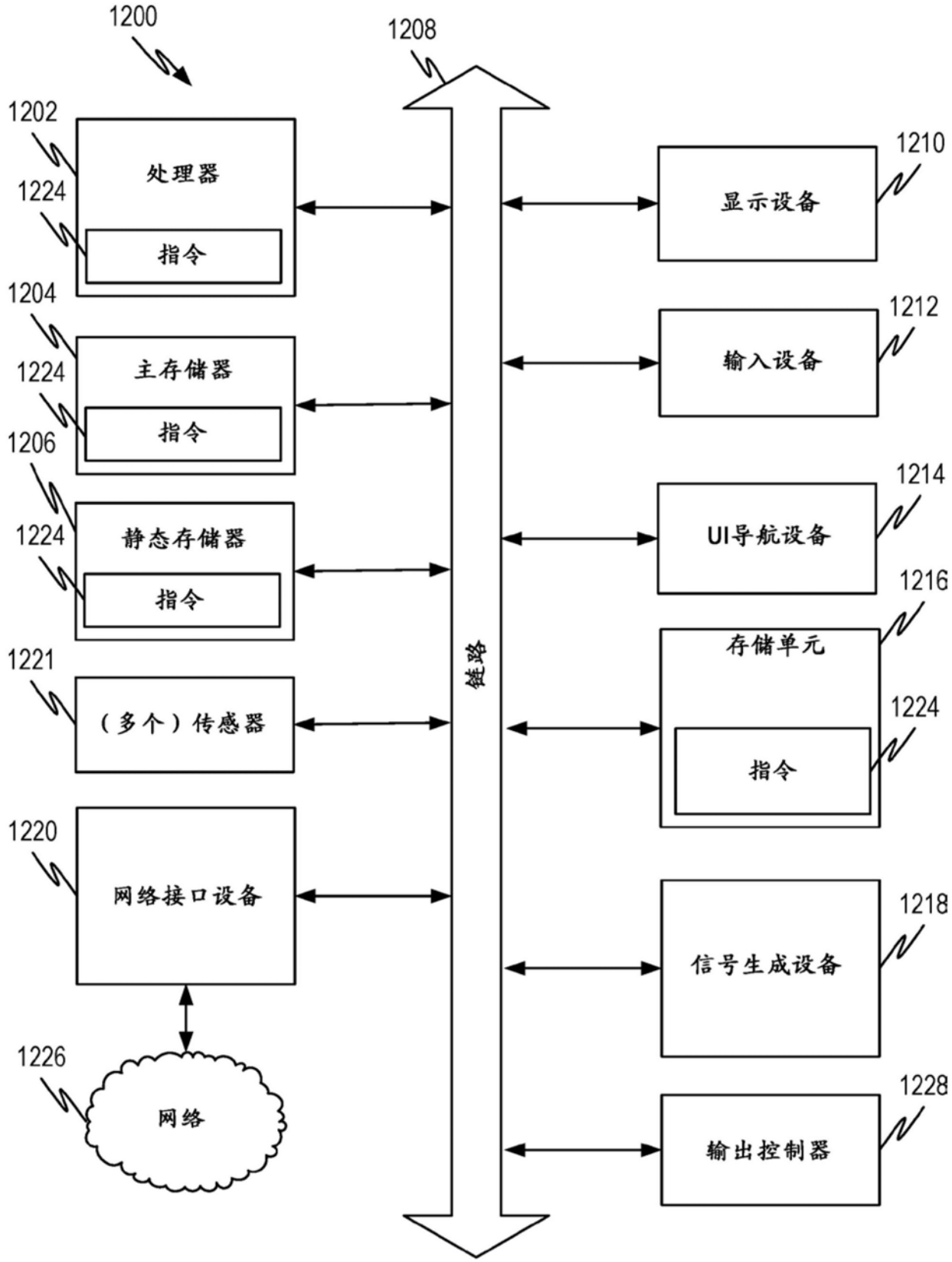

1200
1208
1202
处理器
1224
指令
1204
主存储器
1224
指令
1206
静态存储器
1224
指令
1221
（多个）传感器
1220
网络接口设备
1226
网络
链路
1210
显示设备
1212
输入设备
1214
UI导航设备
1216
存储单元
1224
指令
1218
信号生成设备
1228
输出控制器

图12