SoPware 定义的网络同化:跨越走向集中网络的最后一英里
使用 NAssim 进行配置管理

陈 1，苗玉凯 2，李晨 3，孙海峰 4，洪旭 5，刘立斌 6，龚章 1，魏旺 7，8
1 华为理论实验室，2 新南威尔士大学，3 中关村实验室，
4 北京邮电大学，5 香港中文大学，
6 山东计算机科学中心(位于济南的国家超级计算机中心)，
7 香港科技大学(广州)、8 香港科技大学陈.
huangxun@huawei.com、yukai.miao@unsw.edu.au、lichen@zgclab.edu.cn、HF sun @ bupt . edu . cn @ cuhk . edu . hk、liu.libin@outlook.com、nicholas.zhang@huawei.com、weiw cs@ust.hk

摘要

将新设备加入现有 SDN 网络是网络运营(NetOps)团队的一大难题，因为需要大量专业工作来弥合新设备的配置模型与 SDN 控制器中的统一数据模型之间的差距。在这项工作中，我们提出了一个辅助框架 NAssim，以帮助 NetOps 加速将新设备吸收到 SDN 网络中的过程。我们的解决方案具有一个统一的解析器框架，可将各种设备用户手册解析为初步配置模型；一个严格的验证器，可通过形式语法分析、模型层次结构验证和经验数据验证来确认模型的正确性；以及一个基于深度学习的映射算法，使用最先进的神经语言处理技术，在经过验证的配置模型和 SDN 控制器中的模型之间生成人类可理解的推荐映射。总之，纳西姆解放了大部分的网络操作者
通过直接从设备手册中学习来完成繁琐的任务
SDN 控制器可理解的数据模型
和人类专家。我们的评估显示，NAssim 可以将同化过程加速 9.1 倍。在此过程中，我们还识别并纠正了四家主流供应商设备手册中的 243 个错误，并发布了一个经过验证和专家管理的解析手册语料库数据集，以供未来研究使用。

CCS 概念

•网络→网络可管理性；网络管理；计算方法→机器学习；

这项工作是苗玉凯在华为实习时完成的。通讯作者:陈丽和
孙海峰。

SIGCOMM' 22，2022 年 8 月 22 日至 26 日，荷兰阿姆斯特丹

美国计算机学会国际标准书号 978-1-4503-9420-8/22/08。。。$15.00
https://doi.org/10.1145/3544216.3544244

关键词

多供应商网络、网络配置管理、软件定义的网络

ACM 参考格式:
陈 1，苗玉凯 2，李晨 3，孙海峰 4，洪旭 5，刘立斌 6，龚章 1，魏旺 7，8。2022.软件定义的网络模拟:通过 NAssim 为集中式网络配置管理搭建最后一公里桥梁。2022 年 8 月 22 日至 26 日，荷兰阿姆斯特丹。美国纽约州纽约市 ACM，17 页数。https://doi.org/10.1145/3544216.3544244

1 介绍

软件定义的网络(SDN)是管理现代企业和云网络的流行方法[39,52].SDN 的主要特点是集中控制所有物理和虚拟网络设备。这对于新构建的具有可以运行软件代理的设备的网络来说很容易实现[2,8]可由 SDN 控制器访问。然而，对于具有许多传统设备的网络，当前的 SDN 控制器必须使用它们的命令行界面(CLI)来获得访问。为 SDN 控制器启用基于 CLI 的配置需要网络运营(NetOps)专家付出大量努力，他们必须理解设备的用户手册，找到正确的命令，起草和验证配置模板，并提供将 SDN 配置数据库中的参数映射到模板的规则。除了传统设备，接纳新的供应商并从他们那里引进新的设备也是一项艰巨的任务。由于缺乏用于配置和操作的标准化界面，NetOps 团队需要付出巨大努力，通过开发软件代理，或者如果设备仅具有 CLI 访问权限，则通过上述相同流程，使新设备适应现有的 SDN 控制器。

我们将异构网络设备(例如传统设备和来自新供应商的设备)引入集中控制的现有 SDN 网络的过程定义为软件定义的网络同化(SNA)。我们发现当前的 SNA 方法需要大量的人力，并且容易出错。因此，我们寻求让 SDN 跨越最后一英里，实现所有设备(无论是旧设备还是新设备)的集中配置管理。

SNA 中的关键问题是两个数据模型之间的不匹配:设备的异构配置模型和

统一设备模型(UDM) [16,49]在集中式 SDN 控制器中。我们解决这一问题的方法是从 NetOps 当前的 SNA 实践中学习:为了吸收一个新设备，Ne- tOps 工程师首先通过阅读和理解设备的用户手册来准备命令模板，然后将命令模板中的参数与 UDM 中的参数相关联。为了加速当前的实践，我们的建议 NAssim 也有两个阶段:厂商特定设备模型(VDM)构建阶段，以及 VDM-UDM 映射阶段。在第一阶段，NAssim 帮助 NetOps 工程师从手册中提取并验证 VDM；第二，NAssim 帮助将 VDM 的参数映射到 UDM。

这涉及解决三个挑战:

手册格式异构性:设备的用户手册是提取其配置模型的最可靠来源，尤其是对于只有 CLI 的传统设备。但是，这些手册是为不同的供应商专门组织和格式化的。因此，我们需要一个统一的解析框架来从手册中提取配置模型。

手册中的错误和歧义:尽管手册是真理的唯一来源，但它包含许多错误和歧义，因此仅仅从手册中解析是不够的。为了验证和修正解析模型，我们必须寻求一种有效的方法来识别错误和澄清歧义。

配置模型之间的异构性:不同供应商的配置数据模型是不同的。将它们映射到 UDM 是具有挑战性的，我们认为，必须使用自然语言处理(NLP)技术来学习用户手册中 CLI 命令的自然语言描述。

我们构建 NAssim 作为 NetOps 工程师的辅助框架来应对上述挑战。NAssim 由三个核心组件组成:解析器框架、验证器和映射器。工作流程如下:同化一个新设备，首先 NetOps 工程师使用解析器框架为其用户手册指定一个解析器，运行解析器从手册中提取初步的 VDM；然后，我们使用验证器通过形式分析、模型层次验证和经验数据验证来确认模型的正确性；手册中的任何错误和歧义都在验证器中被捕获，并被汇总和报告给工程师进行纠正；最后，在 VDM 和 SDN 控制器的 UDM 之间，映射器使用新颖的 NLP 算法来产生人类可理解的推荐映射。对于映射中的每个关系，如果 NetOps 工程师发现有问题，NAssim 还可以提供一个最相关的建议列表，其中包含从手册中解析的丰富语义信息，这样他们就不再需要自己搜索手册了。

我们通过以下贡献实现了该工作流:

(1) 我们构建了一个解析器框架来实现网络设备用户手册的定制解析。通过对流行的厂商手册的共性和多样性的综合研究，我们设计了一种与厂商无关的设备模型统一格式，该格式具有可表达性、可解释性和可扩展性。使用这种格式，我们能够进行测试驱动的开发

这增强了设备手册的可靠性

正在解析。根据我们解析主流供应商手册的经验，每个供应商需要 50 行代码。

(2) 由于解析后的结果仍然包含手册中的打字错误或其他人为错误，我们为解析后的结果设计了一个严格且人类可理解的验证框架，该框架有三个阶段:

(a) *形式语法验证:我们将所有解析的命令样式转换成 Backus 范式，并建立相应的语法解析器来验证所有解析命令的一致性。*

(b) *模型层次结构验证:为了验证命令之间的层次结构并识别缺失的命令，我们为每个 CLI 实例构建了一个图模型，并设计了一个基于状态机的模板匹配算法进行验证。*

(c) *经验数据验证:最后，作为一个端到端测试方案，我们使用运行设备的配置来检查解析结果的正确性。我们还提出了一个方案来生成丰富的经验数据，使用上述命令图模型来测试真实设备。*

在我们对 613 个真实配置文件的评估中，我们在配置片段和经验证的模型之间实现了 100%的匹配准确性。NAssim 的验证器还在四个主流供应商的手册中发现了 184 个语法错误和 59 个歧义。

(3) 我们发布了一个经过分析、验证和专家管理的不同供应商的设备手册语料库数据集 1，用于 SDN 和网络管理的未来研究。

(4) 我们采用了最先进的上下文学习 NLP 模型，BERT，并对其进行了扩充，为我们的用例 NetBERT 创建了一个适应领域的版本。我们使用 NetBERT 将任何解析的配置模型映射到 UDM。在将华为和诺基亚的设备型号映射到给定的 UDM 时，NetBERT 分别实现了 89%和 70%的前 10 名召回率。NetBERT 的输出是不同模型参数之间的映射，可由 NetOps 专家理解和编辑。

接下来，我们将概述中的背景 2，并在中展示了 NAssim 的总体设计 3. 然后，我们将在中详细描述它的三个组件 4，5，&6. 我们在 1996 年评估了纳西姆 7,并讨论中的相关工作 9.

注意:就我们所知，这是第一次学习网络配置手册，尽管它在网络管理和操作中很重要。我们发现，作为人类书写的文件，这些手册含有不可忽视的错误，如果盲目遵循，不仅会阻碍 SNA 进程，而且会导致生产问题。纠正这些错误并不简单，需要人类专家的判断和反复试验。这也适用于 VDM-UDM 制图任务。因此，我们认为 SNA 不容易自动化，我们设计 NAssim 作为 NetOps 工程师的辅助 SNA 框架。NAssim 向 NetOps 工程师提供工具和建议，通过自动化最繁琐的任务来加速当前的 SNA 过程，而不是直接提取异构 VDM 并将其映射到 UDM 的端到端系统。

¯https://github.com/AmyWorkspace/nassim

免责声明:这项工作没有提出任何道德问题。

2 背景和动机

在本节中，我们首先强调现代 SDN 网络对 SNA 的需求(§2.1). 接下来，我们总结了 SNA 面临的主要挑战(§2.2). 最后，我们激励和解释 NAssim 的设计决策 (§2.3).

2.1 SDN &对 SNA 的需求

大型企业从多家供应商处采购和运营网络设备。他们还不断将新的设备型号和新的供应商引入他们的网络。这种多供应商的性质使得网络管理任务变得复杂[7,9,20,22, 33,33,34,40,42,44]. 为了减轻管理多厂商网络的难度，NetOps 社区一直依赖 SDN，它为所有网络设备提供了一个逻辑上集中的控制平面。在 SDN 的核心，关键的数据结构是统一设备模型(UDM),它将所有厂商的设备配置模型标准化。结合相关工具[2,8]，SDN 控制器可以配置多供应商设备，就像它们是相同的一样。大型企业和云提供商的网络运营投入了大量精力来构建 UDMs[10, 16, 49].

虽然 UDM 使现有设备的管理变得容易，但它也使新供应商的加入和采用不同配置型号的设备变得困难。由于每个 SDN 网络可能有不同的 UDM，尽管在这方面不断努力，供应商通常没有用于配置和操作的标准化北向接口[8,10,16,49].对于只有 CLI 的传统设备来说，这种情况更糟。对于这些设备，SDN 控制器需要通过 Telnet 连接发送 CLI 命令来实现高级操作意图。构建有效的 CLI 命令还需要 SDN 控制器“理解”CLI 语法，并在 UDM 和设备自身的配置模型之间建立映射。

据我们所知，目前 SNA 完全依赖 Ne- tOps 工程师，他们根据用户手册手工制作新设备和传统设备的模型，并将模型映射到 UDM。整个过程需要大量的人力，并且容易出错。因此，我们寻求一种系统，该系统可以通过以最少的专家努力从手册中学习设备配置模型来加速 SNA，并且可以以人类可理解的方式将所学习的模型映射到 UDM。

2.2 国民账户体系的挑战

为此，我们概述了国民账户体系的三个主要挑战。

手册格式异构性:设备的用户手册通常是获取配置模型最值得信赖的来源，尤其是对于只有 CLI 访问权限的传统设备。手册没有标准化，供应商以不同的方式组织他们的用户手册，如主流供应商的手册所示 4,11,13,15]和附录中的页面快照 A. 尽管风格各异，但它们服务于相同的目的:展示如何通过 CLI 配置设备。因此，用户手册应该涵盖 CLI
设备支持的命令和命令的功能
描述、父视图/工作视图、参数描述和
代表性的例子。然而，同样的概念可能有
每个手册中的不同名称，例如，所有手册都指定了 CLI 命令的部分视图(即，可以接受和执行命令的工作视图)，但可能会将它们
分别在“视图”、“命令模式”、“上下文”和“视图”部分下。我们调查并总结了 most 的五个主要属性
命令参考手册，以及表中四个主流供应商手册的在线版本中相应的级联样式表(CSS)类名 1.

由于主流供应商数量有限，使用 CSS 类名似乎是从在线手册中提取信息的一种微不足道的方式。相反，我们的研究表明，即使对于同一个属性，它在同一个供应商中的 CSS 类名也可能不一致。以思科的在线手册为例。在大多数页面中，它使用“pCE_CmdEnv”类格式化 CLI 命令，而一些页面使用“pCENB _ CmdEnv _ NoBold”。为了区分 CLI 中的占位符参数和关键字，一个手册中的不同页面可以使用来自候选项的一个类，包括“cKeyword”、“cBold”和“cCN_CmdName”。我们怀疑，由于单个设备的手册页数量庞大(每本手册 10k+CLI),手册是在很长一段时间内编写的，并且样式和格式指南会随着时间的推移而改变。类名对于解析的完整性至关重要，否则解析的语料库会丢失手册中的重要信息。然而，收集所有的变体是很费时间的，如果不是不可能的话。

手册中的错误和歧义:我们发现手册包含错误和歧义，盲目地遵循手册会影响 SNA，甚至导致生产问题。例如，我们在 Cisco 手册 2:

```
邻居{ <ip 地址> | <ip 前缀/长度> } [远程 as
  { <as 编号>[<。as-num> ] |路线图<名称> }
```

对于 remote-as 符号前不成对的左括号，有多个潜在的有效选项:去掉左括号，在 remote-as 符号后加一个右括号，在 CLI 末尾加一个右括号；选择哪一个需要专家的判断。然而，这些问题很难被肉眼发现，因此网络运营团队从头到尾审核手册以识别所有问题是不切实际的。

配置模型之间的异构性:每个供应商的配置模型都是不同的，这是有意的
每个供应商设计的配置语言都明显不同于竞争对手的语言，语法也受到专利的严格保护[5].

对于同一概念或意图，不同的供应商使用不同的措辞和语法。桌子 2 显示了 Cisco、Huawei 和 Juniper 的 VLAN 和生成树命令的比较。表中的每一行都显示了每个供应商中用于相同目的的命令。我们可以看到，即使对于简单的任务，配置命令也有很大的不同。因此，将这些不同的配置模型与 UDM 相匹配是一项挑战。

2.3 激励纳西姆的设计决策

我们设计 NAssim 来解决 SNA 的三个挑战。在本节中，我们将概述 NAssim 的设计决策。

[2]https://www.cisco.com/c/en/us/td/docs/switches/datacenter/nexus5500/sw/command/reference/unicast/n5500-ucast-cr/n5500-bgp_cmds_n.html

属性 \ 小贩	华为	加拿大白鲑	诺基亚（总部设在芬兰）	H3C
central library and information services 中央图书馆与信息服务中心	< class="sectiontitle " > 格式	< class="pCE_CmdEnv " >	< class="SyntaxHeader " >语法	< class="Command " >语法
FuncDef	< class="sectiontitle " >函数	< class="pB1_Body1 " >	< class= "描述标题" >描述	< class="Command " >描述
父视图	视图	< class=pCRCM_CmdRefCmdModes > 命令模式	上下文	< class="Command " >视图
游行 f	< class="sectiontitle " >参数	< class="pCRSD_CmdRefSynDesc " > 语法描述	< class="ParametersHeader " >参数	< class="Command " >参数
例子	< class="sectiontitle " >示例	< class="pCRE_CmdRefExample " > 例子	/	< class="Command " >示例

表 1:设备用户手册的多样性:“CLI”字段表示 CLI 命令的正式语法，这些命令是带有占位符参数和特殊字符的命令模板，用于指定选择或可选分支。“ParaDef”字段包含占位符参数的含义和值范围。“FuncDef”字段描述了完整 CLI 的功能。“父视图”字段指示 CLI 的父/工作视图，即一个 CLI 可能有多个可行的工作视图。“示例”字段显示了常见片段的示例，例如，进入父视图并发出实例化的 CLI。

目的	加拿大白鲑	华为	杜松
检查 vlan	显示 vlan [vlanid]	显示 vlan [vlanid]	显示 VLAN-id/VLAN[VLAN id]/[VLAN 名称]
添加/删除 vlan	vlan [vlanid]/无 VLAN[VLAN id]	vlan 分支[vlan id]/撤消 VLAN 分支 [vlanid]	设置 vlan-id [vlanid]/删除 VLAN-id[VLAN id]
配置生成树根桥	生成树 vlan [vlanid]根主服务器	stp 实例[vlanid]根主服务器	生成树 vlan-id [vlanid]根主节点

表 2:思科、华为和 Juniper 的配置语法比较

为了处理多厂商手册中的异构性，我们调查和研究了在线版本的手册，并设计了一个独立于厂商的语料库格式，可以整合不同的手册风格。由于供应商的数量很少(对于典型的网络基础设施招标来说，通常少于 10 个)，并且每个供应商的手动格式对于他们所有的设备来说大致相同，所以我们认为人工构建特定于供应商的解析器是可以接受的。因此，NAssim 的解析器框架侧重于开发过程，并采用测试驱动开发(TDD)方法。对于每个供应商，解析器框架首先基于独立于供应商的语料库格式中的类型限制来生成测试。然后，NetOps 团队可以编写基本的解析组件，并配置 CSS 类名来构建定制的解析器。NAssim 的 TDD 工具为解析器的每个版本生成测试报告，指导开发人员改进解析器。在我们解析主流供应商的经验中，这种人在回路中的工作流既加速了 SNA 过程，又提高了解析语料库的质量。

为了解决第二个挑战，即分析语料库中的错误和歧义，我们在三个级别上执行验证:命令级、命令间级和片段级。对于命令级的正确性，我们采用正式的语法验证技术，并验证解析后的命令是否遵循手册的语法。对于命令间的正确性，我们建立了命令的图模型，并设计了基于状态机的匹配算法来发现和验证命令间的层次关系。对于代码片段级的正确性，我们的创新是包括来自运行设备的配置文件/代码片段，以验证提取的模型，因为确保操作正确性的唯一方法是在实际设备上运行确切的命令。我们使用两个数据源进行验证:1)从运行设备收集的配置文件/片段;
2)对于没有被任何正在运行的设备使用的命令，我们从命令的图形模型中生成配置片段，并通过 CLI 在真实设备上测试它们。

对于最后一个挑战，将提取的模型映射到 UDM，NAs- sim 的创新有两个方面:1)我们选择执行细粒度的参数级映射，而不是命令级或代码段级映射；这样做也导致 VDM 和 UDM 之间的参数到参数映射的人类可理解的输出，这允许网络操作专家直接理解和修改；2)我们使用最近的 NLP 进展来使用所有相关的语义信息生成每个参数的上下文编码，并使用相似性度量来映射它们。当网络操作专家在映射中发现错误时，基于相似性的方法可以为他们节省宝贵的时间，因为 NAssim 可以提供最相关参数的列表以及从手册中解析的丰富语义信息，因此专家不再需要在整个用户手册中搜索此类信息。

3 NASSIM 概述

NAssim 在 SNA 的两个主要阶段帮助 NetOps 工程师:VDM 建设阶段和 VDM-UDM 制图阶段。在这些阶段，我们使用 NAssim 的三个核心组件，如图所示 1.

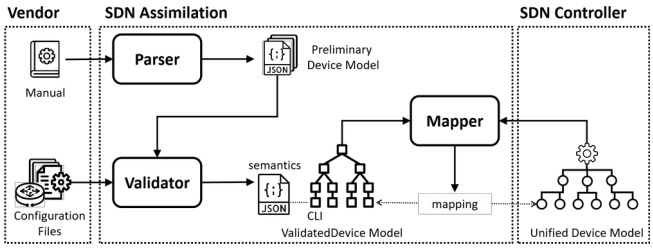


图 1: NAssim 概述。

3.1 VDM 建设阶段

在 VDM 构建阶段，NAssim 的目标是基于其用户手册构建一个经过改进和验证的本地设备模型。我们从描述 VDM 的数据结构开始。然后我们描述了 VDM 构造的关键使能因素:解析器框架和验证器。

VDM 数据结构:我们设计了一个语义增强的树形结构来表示本地设备模型。树的节点是带有占位符参数的 CLI 命令模板，例如节点的对等项
< ipv4 地址>组<组名>”。树的边代表
配置层次结构，例如从节点“bgp <as-number >”到节点“peer < IP v4-address > group < group-name >”的边，表示后一个 CLI 命令在启用的子视图下工作

前任。每个节点被链接到从用户手册中提取的其相关语义语料库，例如节点的对等< IP v4-地址>组
“<组名>”与图中所示的语料库相关联 3. 这
整个树描述了设备的配置模型，包括-
支持的 CLI 命令集、命令层次结构和命令语义。

解析器框架:我们开发了一个解析框架来处理多厂商 CLI 命令语义描述的异构格式，并将它们转换成独立于厂商的格式。如图所示的格式 3 和桌子 3，充当统一的容器，规范手动格式的供应商多样性。NetOps 工程师可以使用这个框架构建定制的特定于供应商的解析器。解析器的输出是初步的 VDM。

验证器:验证器解决了手工的不确定性。它识别初步 VDM 中的缺陷，并生成经验证的 VDM。验证器检查初步 VDM 的三个方面，以执行严格的验证:1)形式语法验证，以识别不明确的 CLI 模板，2) CLI 层次结构推导和验证，以及 3)针对来自运行设备的经验验证配置文件的验证。输出是经过验证的 VDM。

3.2 VDM-UDM 绘图阶段

在这一阶段，NAssim 旨在确定 VDM 和 UDM 之间的参数级关系。

UDM 数据结构:UDM 通常存储为树形层次结构[16].UDM 树的节点表示配置属性，例如接口的 IP 地址、ACL 策略的名称等。属性可以由单个 VDM 中的特定 CLI 命令参数配置，但属性和等效参数可以有不同的名称。子树通常表示一组相关属性，例如特定网络协议的属性。在管理多厂商网络的通用实践中，UDM 涵盖了所有设备的通用功能，这也是多个厂商的 VDM 的交集。在本文中，我们假设 UDM 是给定的，我们将通过为特定供应商的专有 CLI 命令构建 CLI 模型来扩充 UDM 作为未来的工作。构建 UDM 的网络操作工程师通常用属性的简短上下文来注释 UDM，以便于将来的检查和扩展[16].

Mapper: Mapper 旨在解决模型的异构性。即使对于网络操作专家来说，VDM-UDM 映射也是乏味且容易出错的，因为这两种模型的规模都相当大。如表中所示 4，华为和诺基亚的 VDM 都包含超过 104 个节点。Mapper 利用了 VDM 和 UDM 的语义上下文，然后利用上下文编码和相似性度量方面的最新 NLP 进展来实现 VDM-UDM 匹配。映射器的输出是最可能匹配的 CLIs 参数，对于 UDM 上的每个属性具有可解释的语义。我们设计

Mapper 的输出是人类可以理解的:对于 CLI 命令中的每个参数，Mapper 都输出一个建议的列表，这些列表是 UDM 中最相关的属性。NetOps 工程师可以利用他们的领域知识直接处理输出，以进一步确认 VDM-UDM 映射，如果前 1 条建议是错误的，他们可以修改映射结果。我们还收集专家校正的映射结果，并将它们用作带标签的训练/测试集，以持续改进 Mapper 的语义上下文编码和匹配的 NLP 模型，这有利于未来的 VDM-UDM 映射过程。

在接下来的部分中，我们将详细阐述 NAssim 的三个核心组件。

4 NASSIM 解析器框架

手动解析是 SNA 的第一步。我们的解析器框架的主要目标是从手册中提取所有与 VDM 相关的基本信息，同时将不同的手册风格标准化为统一的格式，以便于后续步骤中与供应商无关的处理。为了实现这个目标，我们首先设计了一个统一的格式，充分考虑了表达性、可解释性和可扩展性。然后，我们采用 TDD 方法来保证解析的质量和可靠性。

独立于供应商的 VDM 语料库格式 1, 我们定义了 JSON (JavaScript Object Notation)格式来包含主要的配置信息。数字 3 展示一个由我们的解析框架生成的样本语料库。独立于供应商的格式使用表中的属性组织成字典格式 1 作为钥匙。我们限制每个字段的类型，如表中所示 3. NAssim 的独立于供应商的解析格式捕捉了来自不同供应商的手册的唯一共性，并且很容易向 JSON 扩展额外的信息。这种格式的可读性也使网络操作专家和下游的 NLP 算法变得容易。

键	类型限制
central library and information services 中央图书馆与信息服务中心	字符串列表(非空列表)
FuncDef	线
父视图	字符串列表(非空列表)
游行 f	字典列表(关键字:“段落”和“信息”)
例子	列表列表

表 3:独立于供应商的语料库(JSON)的格式定义

测试驱动的解析器开发:如 2.2, 没有简单的方法来解析手册的在线版本。为了保证解析的完整性，NAssim 的解析框架采用测试驱动开发[29] 原则来指定构建解析器的工作流。

我们的解析框架的架构如图所示 2. 解析器充当解析器基类，由其子类 Parser_<vendor >继承。原则上，我们只需要为每个供应商实现一次 Parser_<vendor>。这个框架便于将一个统一的、可扩展的测试方案容纳到基本解析器类中，以使它的所有子类受益。我们在附录中列出了测试 B.

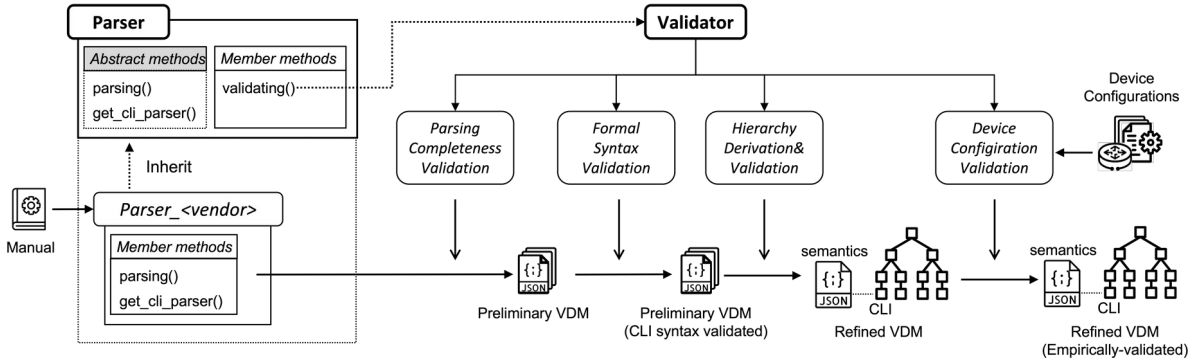


图 2:解析器框架和验证器在 VDM 构建阶段的详细工作流程。

```
{
 "CLIs": [
  "peer <ipv4-address> group <group-name>",
  "undo peer <ipv4-address> group <group-name>"
 ],
 "FuncDef": "The peer group command adds a peer to a peer. The undo peer group
command deletes a peer from a peer group and all configurations of the peer. By default,
no peer group is created",
 "ParentView": ["BGP view"],
 "ParaDef": [
  {"Parameters": "ipv4-address",
   "Info": "Specifies the IPv4 address of a peer. It is in dotted decimal notation."},
  {"Parameters": "group-name",
   "Info": "Specifies the name of a peer group. The name is a string of 1 to 47 case-
sensitive characters, with spaces not supported."}
 ],
 "Examples": [
  ["<HUAWEI> system-view",
   "[~HUAWEI] bgp 100",
   "[*HUAWEI-bgp] group test internal",
   "[*HUAWEI-bgp] peer 10.1.1.1 group test"]
 ]
}

// Extended corpus with distilled ParaType info
{
 "CLIs": [
  "peer <ipv4-address> group <group-name>",
  "undo peer <ipv4-address> group <group-name>"
 ],
 "...": "...",
 "ParaType": {
     "ipv4-address": [ipv4],
     "group-name": [string, 1, 47]}
}
```

图 3:解析后的 VDM 语料库样本。

在我们的解析框架中支持新供应商/模型的工作流程如下:

(0) (可选)我们扩充了基本表 3 带有此供应商的附加类型约束；然后，一个自动化程序生成一组测试；
(1) 我们对一批手册页进行了采样，以开发 Parser_<newvendor >的初步版本，它继承了基本的 Parser 类，并实现了特定的 parsing()方法；
(2) 我们调用在基本解析器中实现的 validating()方法，使用生成的测试来验证解析后的语料库的完整性，并针对违规情况生成一个摘要报告。
(3) 我们在总结报告中跟踪违规情况，以在必要时修改解析逻辑。

我们重复步骤 2 和 3，直到解析的语料库通过所有测试。该报告具有两个部分:1)关键属性的概要，其列出了具有有问题的/空的“CLIs”字段的所有语料库，以及相关的
原始手册页的外部链接；2)语料库的状态，
它列出了每个语料库中所有有问题的字段。开发者可以对最有问题的语料库/页面进行采样，以改进解析器。

由于人工解析位于 SDN 同化的最上游，因此解析的语料库的正确性对于下游任务至关重要。基于 TDD 的人在回路方法的可靠性使它成为我们的场景的一个很好的选择，以一种可解释的方式加速提高解析的语料库的质量。

5 NASSIM 验证器

在解析器框架中，我们引入了 TDD 原则，大大减少了由于粗心解析造成的信息丢失。然而，用户手册是人写的文档，不可避免地会有错误和错别字。原始用户手册中的这些错误信息总是穿透 NAssim 的解析器框架。为了减轻它们对下游任务的负面影响，我们设计了一个严格的验证方案，如图所示 2,包括 CLI 命令的正式语法验证(即
图中的“CLI”字段 3)，配置模型层次结构派生
验证，用经验数据验证。

5.1 形式语法验证

“CLI”字段对错误信息特别敏感，因为格式错误的 CLI 命令无法被接受
网络设备。但是，如中所示 2.2，手册中的 CLI 命令可能包含语法错误，不能完全信任。这促使我们设计一种严格的验证方法，在解析的语料库中系统地审计 CLI 命令，以便专家能够以更有针对性和更有效的方式进行干预。

用户手册的序言提供了说明“CLIs”字段语法的约定。例如，大多数手册用花括号{ }表示选中的分支，用括号[ ]表示可选分支 3，如图所示 4. 我们表达这些
命令约定/语法转换成等效的 Backus Normal

[3]https://www.cisco.com/c/en/us/td/docs/switches/datacenter/nexus5500/sw/command/reference/unicast/n5500-ucast-cr/n5500-ucast-preface.html

Command descriptions use these conventions:

Convention	Description
boldface font	Commands and keywords are in boldface.
italic font	Arguments for which you supply values are in italics.
[ ]	Elements in square brackets are optional.
{x | y | z}	Alternative keywords are grouped in braces and separated by vertical bars.
[x | y | z]	Optional alternative keywords are grouped in brackets and separated by vertical bars.
string	A nonquoted set of characters. Do not use quotation marks around the string or the string will include the quotation marks.

图 4:思科手册的命令惯例。

```
import pyparsing as p

# syntax parser for Cisco CLI
word = p.Word(p.printables, exclude_chars='{}[]|#\n').setParseAction(leaf_gen)
ele = p.Forward()
items = ele + p.ZeroOrMore('|' + ele)
select = p.Group('{' + items + '}').setParseAction(select_gen)
option = p.Group('[' + items + ']').setParseAction(option_gen)
ele <<= p.OneOrMore(option ^ select ^ word).setParseAction(ele_gen)

syntax_parser = ele
```

图 5:语法解析器生成的代码片段。

表格(BNF) [43]，然后通过利用 pyparsing [18]或 ANTLR [1].数字 5 显示了一个 pyparsing 代码片段，用于生成符合图中所示命令约定的语法分析器 4. 为了验证形式语法，我们可以调用语法
解析器以自动方式解析语料库的“CLIs”字段，即检查它们是否符合形式语法。我们记录所有
解析失败案例以快速识别初步 VDM 语料库中有问题的 CLI 字段。然后网络运营专家可以进行有针对性的干预来纠正这些错误，从而提高 VDM 语料库的可信度。

5.2 模型层次结构推导和验证 CLI 模型层次结构指的是各个 CLI 命令之间的关系，即每个命令都有一个可行的工作视图，即由其父 CLI 命令启用的父视图。大多数供应商的配置模型遵循基于树的层次结构，但树通常隐含在手册中
—只有诺基亚的手册指定了它们的层次结构以及单独的 CLI 语法/语义描述。有些设备具有特殊的 CLI 命令，可以
在终端上显示它们的模型层次结构，但只显示命令语法而没有明确链接到它们的语义描述。

通过充分利用 VDM 语料库中的“示例”字段，我们找到了一种可靠的方法来导出伴随语义上下文的模型层次。重新审视图中所示的语料库结构 3，大多数字段以“CLIs”字段为中心，以提供详细的语义
描述。但是,“示例”字段演示了一个实例-
配置代码段中当前“CLI”字段的 tiated 版本，隐式桥接当前 CLI 和之间的关系
它在实例化情况下的父 CLI。因此，我们的层次推导方案的基本思想如下:取图 3 例如，对于一个语料库，我们首先识别相应的 CLI
“示例”字段中的实例，即对等 10.1.1.1 组测试。
然后，我们进行回溯，根据
前缀缩进，即 bgp 100。最后，我们在所有的冲突中搜索，以找到与 CLI 实例 bgp 100 匹配的“CLIs”字段，即在这种情况下的 bgp <as-number >。结合“父视图”字段中指示的“bgp 视图”，CLI 命令 BGP
< as-number >进入“BGP 视图”。具有相同的其他“CLI”

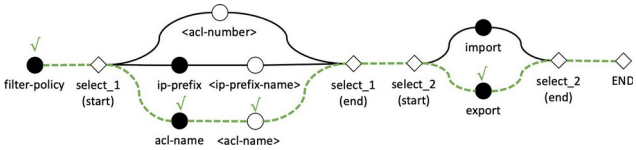


图 6:玩具示例:将 CLI 实例与 CLI 模板的图形模型匹配。

父视图可以直接重用之前的派生，也可以基于自己的“示例”字段进行派生，以便进行多数表决。
CLI 图模型:启用上述层级推导方案(第一和第三步)需要实现一个基本功能:确定 CLI 实例是否匹配 CLI 命令模板，
即“CLIs”字段中的内容。我们设计了一个基于状态机的
此任务的模板匹配算法。我们阐述主要的
玩具示例创意。给定如下 CLI 模板:

过滤策略{ <acl-number >
| ip 前缀< ip 前缀名称>
| ACL-name < ACL-name > } { import | export }

为了检查它是否可以被实例化为特定的 CLI 实例，例如过滤器-策略 ACL-名称 acl1 导出，我们首先构建如图所示的 CLI 图形模型 6. CLI 图模型(CGM)是一个具有单个根和单个终端的有限状态机。然后，我们以广度优先的方式从根节点搜索路径，以匹配 CLI 实例中的令牌。关键字节点(即图中的实心圆 6) 在图中要求精确的文本匹配，而参数节点(即图中的空心圆 6) 仅要求类型匹配，例如 string、int 或 ipv4-addr。如果从根到终端的匹配路径(例如，图中的绿色虚线 6) 找到后，我们可以将 CLI 实例与该模板进行匹配。由于篇幅所限，我们在附录中用例子展示了详细的 CGM 算法 C.

CLI 实例-模板匹配:使用 CGM 作为输入，我们开发了 CLI 实例-模板匹配算法，以确定 CLI 实例是否与 CLI 模板(*cligraph*)匹配，如算法所示 1. 简而言之，我们进行广度优先搜索来找到从根到接收器的路径，以匹配 CLI 实例的令牌。在每个搜索步骤中，我们首先找到下一个元素的所有潜在候选元素。如果都不匹配，匹配算法可以提前终止。否则，我们记录匹配的，并继续下一个搜索步骤。

实际上，CLI 模型层次结构派生方案可以恢复大部分层次结构，但不能保证完全恢复，因为原始手册中可能缺少数据。例如，在华为语料库中，我们发现了相关的例子
“VPN 实例 MSDP 视图”的片段与另一个视图共享
如图所示 7. 对于一个算法来说是困难的和不可靠的
以确定 msdp 命令是启用第一个视图还是第二个视图，或者两者都启用。因此，我们在 CLI 层次结构派生之后引入了一个额外的验证步骤。具体来说，我们基于视图的数量和数据源的例子来量化推导的确定性。对于一个工作视图，如果我们不能可靠地将它与一个示例片段相关联，我们就将它与所有潜在相关的片段一起记录下来，以便网络操作员可以在以后查看它们，用他们的领域知识解决这些不明确的情况。通过大规模自动化推导和目标验证，我们可以

算法 1: CLI 实例-模板匹配

```
Func is_cli_match(cli, cligr aph):Func
    cli_eles = cli.split()
    next_ind = 0
    next = cli_eles[next_ind]
    next _ candis =get_graph_root(cligraph)
    虽然真实如此
        res = match_next(下一个, 下一个_坎蒂斯, cligr aph)
        如果不是 res['match_flag'], 则
          _ 破裂
        next _ candis =get_next_candis(cligraph, res['matched'])
        next_ind += 1
        如果下一个索引> =长度(cli 元素), 则
          _ 破裂
      _ next = cli_eles[next_ind]
    如果下一个索引> = len(cli_eles)且 is_reach_end(next_candis ),则
      _ return {'match_flag': True}
    返回{'match_flag': False}
_
```

```
{
 "CLIs": [
  "import-source acl { acl-number | acl-name }",
  "undo import-source"
 ],
 "...":"... ",
 "ParentView": ["VPN instance MSDP view", "MSDP view of a public network instance"],
  "...":"... ",
 "Examples": [
  ["<HUAWEI> system-view",
   "[~HUAWEI] acl number 3101",
   "...",
   "[*HUAWEI-acl4-advance-3101] quit",
   "[*HUAWEI] multicast routing-enable",
   "[*HUAWEI] msdp",
   "[*HUAWEI-msdp] import-source acl 3101"]
 ]
}
```

图 7:一个不明确视图的例子没有被我们的模型层次结构派生方案解决。

获得带有语义描述的完整 CLI 模型层次结构

单个 CLI 命令，即经验证的 VDM。

5.3 经验数据验证

到目前为止，我们主要利用已解析的 VDM 语料库中的内容进行验证。在这一部分，我们将进一步利用另一个数据源，即经验设备配置。这一验证阶段的工作流程如图所示 8: 对于配置文件中的每个 CLI 实例，我们首先找到其匹配的 CLI 模板；然后，我们检查匹配的模板及其父 CLI 实例的模板是否在 CLI 层次结构上形成父子关系。我们记录不匹配的 CLI 实例和原因(例如，未找到匹配的 CLI 模板、不匹配的层次结构),供专家稍后审核。

我们注意到，现有运行设备上的配置可能不涵盖所有 CLI 命令，因为运行设备可能只启用所需的功能。但是，当前设备配置中包含的 CLI 命令是实践中最常用的命令。

对于那些在经验配置中未使用的 CLI 命令，我们利用构建的 CGM，如图所示 6 生成 CLI 命令实例(例如，枚举从根到汇点的路径并实例化参数节点)；然后，我们将实例直接发送给设备进行验证；最后，我们使用

“show”命令(或非 Cisco 设备上的等效命令)

检查实例是否已在中正确配置

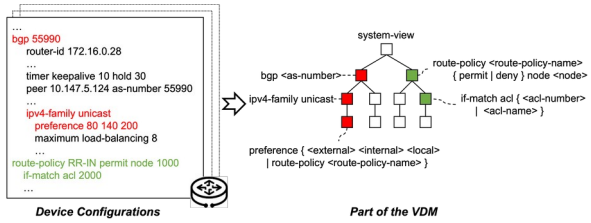


图 8:根据设备配置进行验证。

设备。然后，正确的实例就变成了运行设备的经验配置，我们可以运行上面的相同工作流程。

通过全面的验证方案，我们以 CLI 层次结构和各个 CLI 的语义上下文的形式获得经验证的设备模型，即经验证的 VDM。这为制图者打下了坚实的基础。

6 纳西姆制图仪

由于 CLI 命令和参数的数量庞大，手工制作这样的映射既繁琐又容易出错。因此，我们寻求尽可能自动化这一过程。使用解析器和验证器，VDM 包含所有 CLI 命令及其参数的丰富语义信息，UDM 中的参数也丰富了上下文信息。我们的关键创新是通过应用 NLP 和深度学习(DL)的最新进展，让映射器从 VDM 的上下文信息中“学习”知识，然后预测最有可能映射到 UDM。

问题公式化:本质上，两个设备模型之间的映射是两者之间的模型参数的对齐

模特。给定一个具有nV参数的 VDM V，，$pVPV$=(pV,pV)，

也是一个 UDM UUU 0 1 $UnV-1$

U与nU参数P=($p0$，$p1$，，pnU 1)，映射MV ⊲⊳U是PV和PU之间的二部图。

映射器工作流程:NAssim 映射器的架构如图所示 9. 给定 VDM，我们首先找到每个模型参数的相关 VDM 语料库，然后定位关键上下文信息，例如 CLI 命令及其参数的描述。对于 UDM，我们直接检索每个参数的上下文信息。然后，我们使用基于 DL 的模型对上下文信息进行编码并获得矢量表示，即上下文嵌入。最后，我们通过测量它们的上下文嵌入向量之间的相似性来评估来自两个模型的一对参数是否指同一个概念。我们研究了两个模型中所有可能的参数对，最终

获取映射MV ⊲⊳U。我们在原型中使用的 UDM 是由专家根据他们的经验设计的专有模型

管理大型企业网络的经验。数字 10 显示了将 VDM 的参数映射到 UDM 属性的示例。

6.1 上下文提取

对于每个参数，我们从 VDM(或 UDM)中提取相关的语义信息作为其上下文。形式上，M为 VDM(或 UDM)，与nMparameters 及其参数之一pM (i∈{0，1，，nM1})，我们使用 c(pM)=[s0,s1，,skM1)来表示

iiii

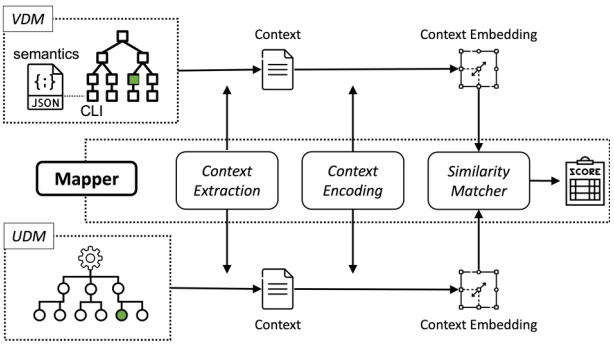


图 9:VDM-UDM 制图中制图者的详细工作流程

我们为piM提取的上下文，其中每个s是一个文本序列和kM是提取的序列数。取决于数量
在每个设备型号的可用信息中，每个型号的kM可能不同。在可用的上下文信息中，我们

找到以下对映射任务有价值的内容:的名称

参数和 CLI 命令、参数描述、父视图以及 CLI 命令的功能描述。

6.2 使用 NetBERT 进行上下文编码/匹配

为了将上下文信息编码成向量，BERT [32]是最流行的适合这个任务的 NLP 模型。给定文本记号的阿瑟序列，BERT 用一堆变换器产生每个记号的上下文化表示 50].SBERT[45]是 BERT 的暹罗网络，它用句子匹配目标和大型 NLI 数据集进行预训练。它能够将两个语义相似的文本序列编码到两个在嵌入空间中接近的嵌入向量中。然而，这样的网络在预训练语料库中从未见过的领域中可能具有有限的性能。

因此，我们采用 SBERT 来完成上下文编码的任务，并将结果模型命名为 NetBERT。NetBERT 是从现有的预训练 SBERT 模型继承来的，该模型通过在映射的 VDM-UDM 参数对上的句子匹配目标进行微调，以实现领域适应(DA)。

嵌入生成:在讨论 DA 的细节之前，我们展示了如何使用 NetBERT 进行上下文编码。我们使用 NetBERT 模型将上下文信息中的文本序列编码成向量表示。我们分别对每个文本进行编码，然后产生一个嵌入矩阵，我们称之为上下文嵌入。假设的输出维度
编码器e是m，那么参数pM的上下文嵌入是
公式化为等式 1.

$$EM_{ii} = e_{iii}(C(pM)) = e([s0, s1, ..., skM-1]) \in RkM \times m \quad (1)$$

如果我们要将 VDM V映射到 UDM U，其中一对参数是(pV, pU)。然后，编码器e()产生一对
上下文嵌入向量(EV_{ij}, EU_{ij}) $\in (RkV \times m, RkU \times m)$，其中$kV$不一定等于$kU$。

参数映射:两个模型参数之间的映射可以通过评估它们对应的上下文嵌入向量的相似性来确定。我们使用行余弦

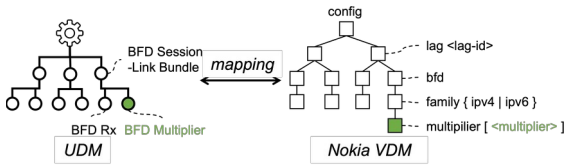


图 10:VDM-UDM 映射的一个例子。

相似性度量两个上下文嵌入向量之间的相似性。对于来自 EV和 EU的每一对行向量，我们
计算余弦相似性得分。给定上下文kV的数量和kU，我们比较了$kV \times kU$向量对。然后，我们建立一个

排列所有的上下文组对，并通过加权和组合所有的余弦分数。等式 2 显示了计算 EV和 EU之间相似性的过程。⊗算出了

两个矩阵的行余弦相似性。

$$Sim(EV_{ij}, EU_{ij}) = w \otimes (EV_{ij}, EU_{ij}) \quad (2)$$

权重矩阵 w 是一个超参数，可以手动指定或通过网格搜索自动生成。在最简单的设置中，我们可以将 w 设置为统一的加权向量，即

$$wt = \frac{1}{kV \times kU} \forall wt$$

我们使用上面的 similarity$V$ m× eUasurement 在 UDM 中查找前 k 个相似参数映射建议。

6.3 微调 NetBERT

给定由 NetOps 专家标记的几个映射的 VDM-UDM 参数对，我们可以生成用于微调 NetBERT 模型的训练语料库。我们将所有映射对视为正对，并进行随机采样以生成负对。对于每一对参数，我们提取它们的上下文，并将它们作为输入输入到当前的 NetBERT 模型中。我们使用与原始 SBERT 模型完全相同的 same 体系结构和句子匹配训练目标来进行微调。在无监督设置的情况下，即没有提供映射的 VDM-UDM，NetBERT 相当于 SBERT。但是通过更多的人力投入，NetBERT 能够更好地适应网络配置领域。

7 估价

我们用 Python 3.8 实现了 NAssim 的原型。在此基础上，我们从以下几个方面对 NAssim 进行了评估:1)VDM 建设阶段的有效性和可靠性；2)VDM-UDM 映射阶段的性能。

7.1 原型实现

对于手动解析框架，我们实现了基础

解析器和解析器_ <供应商>支持解析在线手册-

四大主流厂商的 als (HTML 格式):思科、华为、诺基亚和 H3C。我们主要利用美汤库 3]到 im-
实现它们的 parsing()函数，以及 pyparsing [18]作为解析器

实现 CLI 正式语法分析器的生成器。我们用 1200 行代码(LOC)实现了验证器，我们在模型层次结构的派生和验证(700 LOC)上花了大部分精力，这利用了 NetworkX [14]图书馆。对于映射器，我们使用 PyTorch [19]库来实现模型和算法

供应商/型号/发布年份	-	华为/NE40E/2021	思科/Nexus5500/2011	诺基亚 7750 Sr 2021	H3C/S3600/2009
主要统计数据	#CLI 命令	12874	278	14046	759
	浏览量	607	27	3832	28
	# CLI-视图对	36274	366	22734	851
适应成本	解析()位置	45	52	57	41
	get_cli_parser() LOC	8	6	10	8
语法验证	#无效的 CLI 命令	13	19	139	13
模型层次结构推导和验证	#示例片段	15466	523	/	1147
	建设时间(秒)	785.58	14.29	94.56*	34.3
	#模糊的观点	47	8	/	四
设备配置验证	#配置文件	197	/	416	/
	匹配比率	100%	/	100%	/

表 4:VDM 建设阶段的评估。*诺基亚手册没有提供示例，但他们在手册中明确规定了模型层级。因此，我们使用 Parser_<nokia > 通过实现额外的函数来提取层次结构

800 LOC。我们使用 8 核 3.0GHz CPU、64GB 内存和 1 个 Nvidia V100 GPU 评估了我们在 Ubuntu 18.04 上的实现。

7.2 VDM 建设阶段评价

在这一部分中，我们将回答一个基本问题，即从供应商的用户手册中构建特定于供应商的设备模型是否可行和可靠。因此，我们展示了应用我们提出的解析器和验证器从四个流行供应商的手册中构造精炼和验证的 VDM 的操作经验和结果:华为 NE40E 命令参考[13]，Cisco Nexus 5500 系列 NX-OS 单播路由命令参考[4]，诺基亚 7450 ESS/7750 SR/7950 XRS 参考[15] 和 H3C S3600 命令手册[11].

解析的 VDM 的统计我们首先总结构造的 VDM 的基本统计。桌子 4 显示四个 VDM 中的 CLI 命令总数。如今，一本手册可以涵盖多达 10k+ CLI 命令。在我们的评估中，我们发现单个 CLI 命令在多个视图下工作是很常见的。例如，peer < IP v4-address > as-number < as-number >可以在 BGP 下工作

视图、BGP 多实例视图、BGP-VPN 实例视图等。创建具有指定 AS 号的对等体。在 CLI 模型层次结构中，我们应该使用多个节点来表示具有不同父 CLI 命令的 CLI 命令。换句话说，VDM 的大小应该通过客户端视图对的数量来量化。如表中所示 4，对于所有供应商，CLI-View 对的数量明显大于 CLI 命令的数量。VDM 的巨大规模证明了设计自动化框架的必要性。

使解析器适应新供应商我们接下来调查适应成本。考虑到主要的解析逻辑在所有事件中都是相似的

dors，我们开发了一个解析的基本代码框架。为了实现每个供应商的解析器，我们将特定于供应商的处理细节添加到框架中，主要修改字段的内容提取逻辑

在表中 3. 根据经验，我们使用特定于供应商的解析()对基本框架的修改后的 LOC，以及 get_cli_parser()函数的 LOC 来量化新供应商所需的工作。如我们的评估结果表所示 4，每个供应商需要 50 LOC，这是可接受的一次性成本。

正式语法验证我们应用语法验证方案来识别初步 VDM 中有问题的 CLI 命令。

不足为奇的是，我们在所有供应商中发现了无效的 CLI，如表中所示，总共有 184 个语法错误 4，因为手册毕竟是人写的文档。但是，请注意，无效 CLI 命令占总命令的比例很小。我们认为用户手册作为正式文件应该经历一定的校对和编辑。有了我们的验证提供的基本手动质量保证和修订指导，衍生 VDM 的可信度可以进一步提高。

模型层次验证模型层次推导的计算复杂性取决于视图的数量和示例的数量，如表所示 4. 构建最大的模型层次大约需要 13 分钟，在对示例片段开始 CLI 实例模板匹配之前，大约 84%的处理时间花费在所有 CLI 命令的 CLI 图模型(CGM)生成上。我们认为时间成本是合理的，因为每个手册只做一次。验证器还识别出四个手册中的 59 个不明确的视图，然后报告给 NetOps 进行修改。

经验数据验证在我们的评估中，我们只对华为和诺基亚进行设备配置验证，因为它们的大量配置文件可从数据中心网络获得。我们总共收集了 613 个配置文件，华为 197 个，诺基亚 416 个。华为 set 拥有 93617 行命令实例，其中 17391 行是唯一的。诺基亚集有 163854 行，其中 38352 行是唯一的。我们对照相应的配置文件验证华为和诺基亚的 VDM，如图所示 8.我们发现配置文件中 100%的 CLI 实例可以匹配到 CLI 模型层次结构上的节点，这证实了我们的层次结构推导方案的完整性。在我们的评估中，华为数据集仅与 153 个命令模板相关联，与总共 12874 个模板相比，这是很少的。这证实了该数据集来自数据中心的事实，在数据中心中，成千上万的设备使用相同的功能集。

7.3 VDM-UDM 制图阶段的评估

我们继续回答这个问题:提议的映射器能减轻 SNA 中网络操作工程师的负担吗？如前所述，映射器充当推荐系统，生成 VDM 和 UDM 参数之间最可能的映射。因此，我们评估

映射设置	模型	召回中的k @前k (%)							
		一	3	5	七	9	10	20	30
华为-UDM	红外辐射（Infrared Radiation）	41	61	69	76	79	80	90	93
	希姆策	40	59	66	68	70	72	77	81
	斯贝特	53	72	79	81	84	85	89	92
	IR+SimCSE	43	68	75	79	81	82	89	92
	IR+SBERT	56	75	81	85	87	88	91	94
	奈特伯特	57	74	80	85	86	87	91	94
	IR+NetBERT	**58**	**78**	**83**	**86**	**88**	**89**	**93**	**95**
诺基亚-UDM	红外辐射（Infrared Radiation）	24	41	48	57	59	60	68	70
	希姆策	20	31	37	39	39	42	48	53
	斯贝特	34	38	49	49	52	52	58	63
	IR+SimCSE	24	35	42	46	48	48	57	61

表 5:映射器性能

Mapper 的映射性能，以量化其对网络运营的好处。

度量:为了评估映射器的映射性能，我们采用 Recall@top-k，它表示正确匹配参数在映射器的 top k 建议中的测试用例的百分比。k 越小，召回率越高，意味着映射器对协助网络操作更有帮助。

事实:为了评估映射器，我们要求 NetOps 工程师帮助将华为和诺基亚的 VDM 标注到给定的 UDM。我们总共有 381 个 UDM 和华为 VDM 之间的映射注释，以及 110 个诺基亚的映射注释。

微调:考虑到带注释的数据很少，我们使用跨供应商调优和验证来评估微调的有效性。我们使用来自诺基亚的带注释的参数对微调 NetBERT，然后在华为数据集上评估其性能。同样，我们在华为数据集上微调 NetBERT，然后在诺基亚数据集上进行评估。我们在阴性抽样中采用 1:10 的阳性/阴性比率。我们观察到只有 1 个时期的训练是必要的，因为更多的时期可能容易导致过度拟合。

对比模型:我们对比这些模型:

*IR:一个自然的想法是使用信息检索(IR)方法来生成候选映射。我们使用 TF- IDF [*38,41]测量 UDM 和 VDM 之间参数对的相似性，并报告得分最高的 k 个参数对。

*SBERT [*45]使用连体和三元组网络结构增强预训练的 BERT 网络，以导出可使用余弦相似度进行比较的有意义的句子嵌入。

*SimCSE: SimCSE [*35]利用对比学习目标来正则化预先训练的嵌入，以利于句子相似性测量任务。

IR+DL:该模型集成了 IR 方法和 DL 模型，其中 DL 可以是任何基于 DL 的句子匹配模型。他们首先利用 IR 方法粗略匹配前 50 名候选人，然后利用 DL 模型对候选人进行精细排序，以输出最终的前 k 名匹配候选人。

结果:我们在 VDM-UDM 映射任务上评估了不同语境理解模型的回忆@top k，k 从 1 到 30。结果总结在表中 5. 在无监督设置下，SBERT 的表现始终优于 SimCSE。复合 IR+DL 模型实现了比 ir 或 DL 模型更好的性能。当提供监督时，微调的 NetBERT 优于所有其他模型，这是因为我们使 NetBERT 模型适应网络配置的领域。对于华为数据集，在所有 top-k 指标上，微调的改进是 1-4%。对于诺基亚数据集，微调实现了高达 18%的改善(召回@10)。这是因为华为数据集比诺基亚数据集有更多的训练对，因此对华为数据集的微调更有效。此外，我们观察到 NetBERT 的性能与 IR+NetBERT 一样好，甚至在某些 recall@k 中超过了它。这表明一个良好调整的 NetBERT 模型足以完成我们的任务。关于节省人力，对于华为，网络运营工程师可以在 NetBERT 的前 10 个建议中找到正确的参数对，准确率为 89%。这意味着，如果允许 Mapper 为参数对匹配提供 10 条建议，则 NetOps 工程师在映射阶段只需用 11%的时间参考手册，从而将映射阶段的速度提高了 9.1 倍。

8 讨论

8.1 设备配置模型

网络设备常见的配置模型有两种:CLI 和 YANG/NETCONF。CLI 是配置网络设备的入门级方法。现有基础设施中的几乎所有设备都支持 CLI 功能，无论是最新的还是传统的，而且几乎所有供应商都在其手册中提供了对 CLI 命令的全面描述。因此，通过 CLI 配置网络是很直观的。YANG/NETCONF 被认为是一种先进的网络配置方式。YANG 是一种用于 NETCONF 配置管理协议的数据建模语言。YANG/NETCONF 的目标是以更加结构化的方式实现配置数据的推送和拉取，以实现网络自动化。然而，杨/NETCONF 比 CLI 更难掌握，因为它的复杂性更高。杨描述了网络的节点及其相互作用。每个 YANG 模块都定义了一个数据层次结构，可用于基于 NETCONF 的操作。模块可以从其他外部模块导入数据，并包含来自子模块的数据。虽然 YANG 模式确实是一个标准，但是 YANG 模型中表示的数据会根据供应商的实现而变化。我们在实地采访中发现，掌握杨/NETCONF 的网络运维专家远远少于掌握 CLI 的网络运维专家。如设备供应商维护的储存库中所示[23 – 25]，特定于供应商的 YANG 模型不如手册中详述的 CLI 模型直观[4,13,15]. 在这项工作中，我们采用基于 CLI 的特定于供应商的设备模型(VDM)作为基础，以便 SDN 同化可以覆盖更多的供应商和设备，无论是旧的还是新的；在这个阶段，NAssim 可以被更多的网络运营团队更顺利地采用，以便持续改进我们的方法。据信，NAssim 的核心“解析-验证-映射”哲学也可用于解决特定于供应商的 YANG 模型和统一设备模型(UDM)之间的异构性，但应做出额外努力来总结 YANG 知识库中的直观表示[23 – 25].

8.2 供应商中立参考框架

几十年来，随着多厂商企业级网络的日益复杂，网络运营部门一直试图减轻网络配置的负担。杨的发展是最具代表性的努力。杨在2010年根据RFC 6020 [28]并且目前由互联网工程任务组(IETF)维护。IETF YANG试图创建厂商中立的数据模型。然而，它在范围上仍然受到限制，因为它可以在多个供应商之间进行配置，并且被广泛认为是最容易开始的YANG版本。除此之外，供应商创建了他们自己版本的YANG来控制特定于他们设备的特性[23 – 25].这使得管理多厂商网络变得困难，因为需要为不同的设备创建和维护不同的脚本。OpenConfig项目[16]来创建一个在多个供应商之间可互操作的供应商中立的参考模型。如今，像思科这样的一些厂商支持OpenConfig YANG。然而，他们的本地数据模型仍然提供了大部分的配置覆盖，而他们支持的开放模型只覆盖了一小部分特性[26].OpenConfig要实现完全的互操作性还有很长的路要走。回顾杨的发展历程，不难理解，硬件厂商为了增强竞争力，不断推出新的功能。这无疑有助于提高网络性能，但也在生态系统中造成了进一步的分裂，使得构建供应商中立的配置参考框架变得更加困难。NAssim从实用的角度出发，初步尝试吸收多家厂商的设备型号。然而，要使我们的网络朝着更易管理的基础设施发展，还需要学术界和社区做出更大的努力。

8.3 超越SDN的网络同化

NAssim的设计受到网络配置管理中的实际挑战的激励。大多数企业级网络呈现多厂商性质，并且在当前网络操作实践中，在上层网络功能(北向)和异构网络设备(南向)之间普遍存在软件定义的控制器。SDN控制器的南行使得异构设备对于北行是透明的。例如，如果网络功能需要改变BGP协议的自治系统号，则控制器应该执行正确的配置命令，以使改变在目标设备上生效，而不管它们的供应商。目前，将多厂商设备配置模型融入中央控制器需要大量的人力。因此，NAssim寻求使这一过程对网络作战来说更有效、更经济。从更广的角度来看，网络同化的基本研究问题是确定网络设备之间的语义对等。NAssim是在软件定义的网络集中式配置管理环境中解决这一问题的初步尝试。简而言之，NAssim旨在识别控制器中多供应商设备模型和统一设备模型之间具有等效配置效果的命令和参数。在网络自动驾驶的设想中，应该做出更多的努力来识别配置命令之外的语义等价的其他方面，以及传统路由器/交换网络之外的其他网络范例，即语义互操作物联网网络中的能力[27].

9 相关作品

据我们所知，我们对网络设备手册进行了第一次全面的研究和验证，NAssim是第一项旨在通过从手册中解析配置模型来简化引入新供应商和设备的流程的工作。

最近的工作[10,16,46 – 49]建议使用集中式udm，如OpenConfig、FBNet进行网络控制和管理。几种工业解决方案[12,17,21,40]采用基于模板的方法来生成配置。许多努力还旨在开发抽象语言或模型，以厂商中立的方式指定配置[6,16,49]或以方便用户的方式[37].然而，没有人考虑从半结构化用户手册解析配置模型的问题，因此对于这些SDN网络，吸收新设备需要大量的人力。

另一类相关工作是网络验证，它通过正式方法在提供商网络和企业网络中进行分析30,31,34,36,51].然而，它们仅适用于有限的供应商，并且不考虑手册中的错误和不一致。NAssim可以帮助将它们扩展到其他供应商，这些供应商不容易使用经过解析和验证的语料库和映射模型提供支持。最近的自然语言处理技术采用深度预训练的语言模型将文本序列编码成高维嵌入向量，该向量携带其语义。伯特[32]是最流行的预训练语言模型，它已经被应用于各种自然语言处理任务中。虽然BERT擅长捕获训练语料的语义信息，但通常需要进行微调以适应下游任务。SBERT[45]是一个专门为句子匹配而预先训练的暹罗BERT网络，它自然符合我们的需要。SimCSE[35]是另一个基于BERT的句子匹配模型，它是用对比学习目标预先训练的。我们在映射任务上评估了SBERT和SimCSE，发现它们的性能低于NetBERT，后者是我们经过微调和领域适应的模型。

10 结论

当前的SNA流程是NetOps工程师的一大难题，需要投入大量人力来阅读和理解手册，以构建新设备的配置模型并将其连接到SDN控制器中的UDM。我们建立NAssim是为了帮助和加速SNA进程。我们的解决方案具有一个统一的解析器框架、一个严格的验证器和一个使用领域自适应BERT模型的映射器。NAssim通过直接从手册中学习，形成SDN控制器和人类专家都能理解的设备模型，解放了网络操作工程师。我们的评估显示，我们可以将同化过程加速9.1倍。我们还发布了一个经过验证和专家管理的解析人工语料库数据集，供未来研究使用。

鸣谢:我们感谢匿名SIGCOMM观众的建设性反馈和建议。本工作得到了国家重点R&D项目2020YFB1807800、香港研究资助局(11209520)和CUHK (4055138，4937007，4937008，
5501329，5501517).

参考

[1] 在线；最后一次访问时间是2022年1月。另一种语言识别工具。https://www.antlr.org/。(2022).

[2] 在线；最后一次访问时间是2022年1月。阿普斯特拉。https://apstra.com/.(2022).

[3] 在线；最后一次访问时间是2022年1月。漂亮-汤库。https://beautiful-soup-4.readthedocs.io/en/latest/.(2022).

[4] 在线；最后一次访问时间是2022年1月。思科Nexus 5500系列NX-OS单播路由命令参考。https://www.cisco.com/c/en/us/td/docs/switches/数据中心/nexus 5500/SW/command/reference/unicast/n 5500-ucast-Cr . html。(2022).

[5] 在线；最后一次访问时间是2022年1月。思科就知识产权起诉华为 。 https://www.computerworld.com/article/2578617/cisco-sues-huawei-over-intellectual-property.html。(2022).

[6] 在线；最后一次访问时间是2022年1月。分布式管理任务组公司。https://www.dmtf.org/。(2022).

[7] 在线；最后一次访问时间是2022年1月。脸书、Tinder、Instagram遭遇广泛问题。https://Mashable . com/2015/01/27/Facebook-tinder-insta gram-issues/。(2022).

[8] 在线；最后一次访问时间是2022年1月。gNXI工具- gRPC网络管理/操作接口工具。https://github.com/google/gnxi/.(2022).

[9] 在线；最后一次访问时间是2022年1月。谷歌路由失误周五让日本互联网陷入黑暗。https://www.theregister.co.uk/2017/08/27/google_routing_blunder_sent_japans_internet_dark/。(2022).

[10] 在线；最后一次访问时间是2022年1月。AOS基于图的实时查询。https://apstra.com/products/。(2022).

[11] 在线；最后一次访问时间是2022年1月。H3C S3600命令手册。http://www.h3c.com/en/Support/Resource_Center/HK/Switches/H3C_S3600/H3C_S3600_Series_Switches/Technical_Documents/Command/Command/H3C _ s 3600 _ CM-Release _ 1602(v 1.02)/。(2022).

[12] 在线；最后一次访问时间是2022年1月。HPE网络管理(惠普OpenView)。https://www8.hp.com/us/en/solutions/business-solutions/printingsolutions/overview.html。(2022).

[13] 在线；最后一次访问时间是2022年1月。华为NE40E命令参考。https://support.huawei.com/enterprise/en/routers/ne40e-pid-15837?类别=参考指南。(2022).

[14] 在线；最后一次访问时间是2022年1月。NetworkX库。https://networkx.org/,.(2022).

[15] 在线；最后一次访问时间是2022年1月。诺基亚7450 ESS/7750 SR/7950 XRS参考。https://infocenter.nokia.com/public/7750SR140R4/index.jsp.(2022).

[16] 在线；最后一次访问时间是2022年1月。OpenConfig。http://openconfig.net/.(2022).

[17] 在线；最后一次访问时间是2022年1月。Opsware。http://www.opsware.com/.(2022).

[18] 在线；最后一次访问时间是2022年1月。pyparsing模块。https://pyparsing-docs.readthedocs.io/en/latest/index.html.(2022).

[19] 在线；最后一次访问时间是2022年1月。PyTorch图书馆。https://pytorch.org/.(2022).

[20] 在线；最后一次访问时间是2022年1月。在小故障导致大停电后，纽约证券交易所股票交易停止。https://www.theguardian.com/business/live/2015/jul/08/纽约证券交易所华尔街。(2022).

[21] 在线；最后一次访问时间是2022年1月。Tivoli Netcool配置管理器。http://IBM . com/software/products/en/tivonetconfmana。(2022).

[22] 在线；最后一次访问时间是2022年1月。联合航空公司的飞机因电脑路由器故障而停飞。https://www.bbc.com/news/technology-33449693.(2022).

[23] 在线；最近一次访问是在六月。2022.思科杨模型库。https://github.com/yang models/yang/tree/main/vendor/Cisco。(2022).

[24] 在线；最近一次访问是在六月。2022.华为杨模型库。https://github . com/华为/杨。(2022).

[25] 在线；最近一次访问是在六月。2022.杨模型库。https://github.com/nokia/7x50_YangModels。(2022).

[26] 在线；最近一次访问是在六月。2022.Cisco中的OpenConfig支持。https://www.ciscolive.com/c/dam/r/ciscolive/us/docs/2019/pdf/DEVNET-1775.pdf。(2022).

[27] 在线；最近一次访问是在六月。2022.什么是物联网中的语义互操作性，为什么它很重要？https://www.ericsson.com/en/blog/2020/7/物联网语义互操作性。(2022).

[28] 在线；最近一次访问是在六月。2022.一种用于网络配置协议(NETCONF)的数据建模语言。https://datatracker.ietf.org/doc/html/rfc6020。(2022).

[29] 肯特·贝克。2003.测试驱动开发:举例。艾迪生-卫斯理专业。

[30] 瑞安·贝克特、阿尔蒂·古普塔、拉图尔·马哈詹和大卫·沃克。2017.网络配置验证的一般方法。进行中。ACM信号通信。

[31] 瑞安·贝克特、拉图尔·马哈詹、托德·米尔斯坦、吉坦德拉·帕德伊和大卫·沃克。
2016.不要介意差距:桥接网络范围的目标和设备级配置。进行中。ACM信号通信。

[32] 雅各布·德夫林、张明蔚、肯顿·李和克里斯蒂娜·图塔诺娃。2019.伯特:用于语言理解的深度双向转换器的预训练。在NAACL。计算语言学协会，明尼苏达州明尼阿波利斯，4171–4186。https://doi.org/10.18653/v1/N19-1423

[33] Ahmed El-Hassany、Petar Tsankov、Laurent Vanbever和Martin Vechev。2018.NetComplete:具有自动完成功能的实用全网配置综合。进行中。USENIX·NSDI。

[34] 阿里·福格尔、冯淬帆、路易斯·佩德罗萨、梅格·瓦尔拉德·沙利文、拉梅什·戈文丹、
拉图尔·马哈詹和托德·米尔斯坦。2015.网络结构分析的一般方法。在Proc USENIX NSDI。

[35] 高天宇、姚兴成和陈。2021.简单对比
句子嵌入的学习。(2021).

[36] Aaron Gember-Jacobson，Wu，，Aditya Akella和Ratul Maha-2015年1月。管理平面分析。进行中。ACM IMC。

[37] 阿瑟·雅各布斯、里卡多·普菲策尔、拉斐尔·里贝罗、罗纳尔多·费雷拉、
利桑德罗·z·格兰维尔、沃尔特·威林格和桑杰·g·拉奥。2021.嘿，米露！使用自然语言进行基于意图的网络管理。2021年USENIX年度技术会议(USENIX ATC 21)。USENIX协会，625-639。https://www.usenix.org/conference/atc21/presentation/jacobs

[38] 凯伦·斯帕克·琼斯。2004.术语特异性的统计解释及其在检索中的应用。j .第60号文件(2004年)，第493-502页。

[39] 金孝俊和尼克·费斯特。2013.通过以下方式改善网络管理
软件定义的网络。IEEE通信杂志51，2 (2013)，114-119.

[40] 刘、、、、、、雷洲、马青和张明。2018.NetCraft:网络配置的自动生命周期管理。进行中。ACM SelfDN。

[41] 汉斯·彼得·鲁恩。1957.机械化编码的统计方法
文学信息检索。IBM研发中心。1 (1957)，309–317.

[42] 拉托尔·马哈詹、大卫·韦瑟罗尔和汤姆·安德森。2002.了解BGP错误配置。进行中。ACM信号通信。

[43] 丹尼尔·麦克拉肯和埃德温·赖利。2003.巴克斯-诺尔形式。在…里
计算机科学百科全书。129–131.

[44] 瞻博网络。技术报告，2018年5月。网络宕机背后的原因是什么？减少人为错误和提高网络可用性的积极措施。(技术报告，2018年5月)。

[45] 尼尔斯·雷默斯和伊琳娜·古雷维奇。2019.句子伯特:使用暹罗伯特网络的句子嵌入。在EMNLP。计算逻辑协会，中国香港，3982–3992。https://doi.org/10.18653/v1/D19-1410

[46] 布兰登·施林克、拉迪卡·尼兰詹·迈索尔、肖恩·史密斯、杰弗里·C·莫古尔、阿明·瓦达特、于敏兰、伊桑·卡茨-巴塞特和迈克尔·鲁宾。2015.Condor:通过声明式设计实现更好的拓扑。进行中。ACM信号通信。

[47] 孙欣和杰弗里·克谢。2013.通过以下方式最大限度降低网络复杂性
集成自上而下的设计。进行中。ACM上下文。

[48] 宋玉伟，，桑杰. G .饶，杰弗里. G .谢和大卫. A .马尔茨。2010.走向企业网络的系统化设计。IEEE/ACM网络汇刊19，3 (2010)，695–708。

[49] 宋玉伟，铁，黄海燕和曾。2016.Robotron:脸书规模的自顶向下网络管理。进行中。ACM信号通信。

[50] Ashish Vaswani、Noam Shazeer、Niki Parmar、Jakob Uszkoreit、Llion Jones、Aidan N. Gomez、ukasz Kaiser和Illia Polosukhin。2017.你需要的只是关注。在NIPS (NIPS'17)。美国纽约州红钩市柯伦联合有限公司，邮编6000–6010。

[51] 康斯坦丁·韦茨、道格·沃斯、艾米娜·托拉克、迈克尔·恩斯特、阿尔温德·克里希-
纳穆蒂和扎卡里·塔洛克。2016.使用SMT解算器对边界网关协议配置进行可扩展验证。进行中。ACM OOPSLA。

[52] 夏文峰，温永刚，传恒福，杜斯特尼亚托，谢海勇。
2014.软件定义网络综述。IEEE通信调查与教程17，1 (2014)，27–51。

附录

附录是未经同行评审的支持材料。

A 手动快照

我们在图中显示了思科、诺基亚、华为和 H3C 在线版手册页的屏幕截图 11, 12, 13, 14, 分别是。

redistribute (BGP)

To inject routes from one routing domain into the Border Gateway Protocol (BGP), use the **redistribute** command. To remove the **redistribute** command from the configuration file and restore the system to its default condition in which the software does not redistribute routes, use the **no** form of this command.

redistribute {**direct** | **eigrp** *instance-tag* | **ospf** *instance-tag* | **rip** *instance-tag* | **static**} [**route-map** *map-name*]

no redistribute {{**direct** | **eigrp** *instance-tag* | **ospf** *instance-tag* | **rip** *instance-tag* | **static**} [**route-map** *map-name*]

Syntax Description

direct	Distributes routes that are directly connected on an interface.
eigrp *instance-tag*	Specifies the name of an EIGRP instance. The *instance-tag* can be any case-sensitive, alphanumeric string up to 20 characters.
ospf *instance-tag*	Distributes routes from the OSPF protocol. This protocol is supported in the IPv4 address family. The *instance-tag* can be any case-sensitive, alphanumeric string up to 20 characters.
rip *instance-tag*	Distributes routes from the RIP protocol. The *instance-tag* can be any case-sensitive, alphanumeric string up to 20 characters.
static	Redistributes IP static routes.
route-map *map-name*	(Optional) Specifies the identifier of a configured route map. Use a route map to filter which routes are redistributed into EIGRP.

Command Default

Disabled

Command Modes

Command HistoryAddress family configuration mode
Router configuration mode
Router VRF configuration mode

Release	Modification
5.0(3)N1(1)	This command was introduced.

Usage Guidelines

Use the **redistribute** command to import routes from other routing protocols into BGP. You should always use a route map to filter these routes to ensure that BGP redistributes only the routes that you intend to redistribute.

You must configure a default metric to redistribute routes from another protocol into BGP. You can configure the default metric with the **default-metric** command or with the route map configured with the **redistribute** command.

This command requires the LAN Enterprise Services license.

Examples

This example shows how to redistribute BGP routes into an EIGRP autonomous system:

```
switch(config)# router bgp 64496
switch(config-router) address-family ipv4 unicast
switch(config-router-af)# redistribute eigrp 100
```

Related Commands

Command	Description
default-metric (BGP)	Sets the default metrics for routes redistributed into BGP.

图 11:思科手册快照。

B 解析完整性验证测试

在我们的实践中，我们特别实施了以下验证测试，以确保解析质量。

键完整性测试:解析后的 JSON 文件应该是一个字典，表中至少有五个基本键: CLIs、FuncDef、ParentViews、ParaDef 和 Examples。

*类型限制测试:每个字段都应符合类型*表中定义的限制 3.

CLI 关键字/参数自检测试:我们对关键的“CLI”字段进行额外检查，以确保 CLI 关键字/参数识别的正确性。在原始 HTML 中

label-ipv4

Syntax

label-ipv4 send *send-limit* **receive [none]**
label-ipv4 send *send-limit*
no label-ipv4

Context

config>router>bgp>add-paths
config>router>bgp>group>add-paths
config>router>bgp>group>neighbor>add-paths

Description

This command is used to configure the add-paths capability for labeled-unicast IPv4 routes. By default, add-paths is not enabled for labeled-unicast IPv4 routes.

The maximum number of labeled-unicast paths per IPv4 prefix to send is the configured *send-limit*, which is a mandatory parameter. The capability to receive multiple labeled-unicast paths per prefix from a peer is configurable using the **receive** keyword, which is optional. If the **receive** keyword is not included in the command, receive capability is enabled by default.

The **no** form of the command disables add-paths support for labeled-unicast IPv4 routes, causing sessions established using add-paths for labeled-unicast IPv4 to go down and come back up without the add-paths capability.

Default

no label-ipv4

Parameters

send-limit— The maximum number of paths per labeled-unicast IPv4 prefix that are allowed to be advertised to add-paths peers. (The actual number of advertised routes may be less.) If the value is none, the router does not negotiate the send capability with respect to label-IPv4 AFI/SAFI.
Values— 1 to 16, none

receive — The router negotiates to receive multiple labeled-unicast routes per IPv4 prefix.

none— The router does not negotiate to receive multiple labeled-unicast routes per IPv4 prefix.

图 12:诺基亚手册的快照。

peer as-number (BGP view)

Function

The **peer as-number** command creates a peer and configures an AS number for a specified peer.
The **undo peer as-number** command deletes the AS number of a specified peer.
By default, no BGP peer is configured, and no AS number is specified for a peer.

Format

peer *ipv4-address* **as-number** *as-number*
undo peer *ipv4-address*

Parameters

Parameter	Description	Value
ipv4-address	Specifies the IPv4 address of a peer.	It is in dotted decimal notation.
as-number	Specifies a destination AS number.	For an integral AS number, the value is an integer ranging from 1 to 4294967295. For an AS number in dotted notation, the value is in the format of x.y, where x and y are integers ranging from 1 to 65535 and from 0 to 65535, respectively.

Views

BGP view

Default Level

2: Configuration level

Task Name and Operations

Task Name	Operations
bgp	write

Usage Guidelines

Usage Scenario
The **peer as-number** command is used to create a BGP peer.
Precautions
If a peer does not join any peer group or the peer group to which a peer belongs is not configured with an AS number, deleting the AS number of the peer will reset the peer relationship.
If a peer in a peer group is not configured with an AS number, deleting the AS number of the peer group will interrupt the connection on the peer.
The AS number for external session group cannot be the same as the local AS number.
If you run the **undo peer ipv4-address** command, all configurations related to the peer are deleted. Therefore, exercise caution when running this command.

Example

Set the AS number to 100 for IPv4 peer 10.1.1.1.

```
<HUAWEI> system-view
[~HUAWEI] bgp 100
[*HUAWEI-bgp] peer 10.1.1.1 as-number 100
```

图 13:华为手册快照。

具有富文本格式(RTF)、关键字和参数的页面通常通过它们的字体格式来区分。通过我们的解析框架解析器，参数应该在纯文本中用尖括号表示，如图所示 3.

ospf cost

Syntax

ospf cost *value*

undo ospf cost

View

Interface view

Parameters

value: Cost for running an OSPF process on an interface, in the range of 1 to 65535.

Description

Use the ospf cost command to configure the OSPF cost on an interface.

Use the undo ospf cost command to restore the default.

By default, the OSPF cost on an interface is 10.

You can use the display ospf brief command to display the OSPF cost information.

Related commands: display ospf brief.

Examples

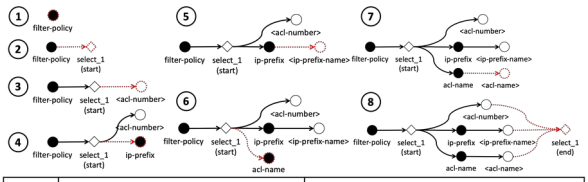
Specify the OSPF cost on the interface as 33.

```
<Sysname> system-view
System View: return to User View with Ctrl+Z.
[Sysname] interface Vlan-interface 10
[Sysname-Vlan-interface10] ospf cost 33
```

图 14:H3C 手册的快照。

为了进行 CLI 关键字/参数自检测试，我们提取了参数标记，即
然后通过将这些令牌与“ParaDef”字段中的参数进行比较来进行交叉检查。通过我们的验证测试，我们很快发现思科手册 inter-
可变地使用“cKeyword”、“cBold”和“cCN _ CmdName”CSS 标签来表示 CLI 命令中的关键字，华为手册可互换地使用“CmdName”和“strong”。在测试驱动开发的帮助下，我们通常可以
一天之内开发一个特定的解析器_<vendor >。

C CGM 算法详细信息



State	Before insertion		After insertion	
	prev_stack	tail_stack	prev_stack	tail_stack
1	[ ]	[ ]	[filter-policy]	[filter-policy]
2	[filter-policy]	[filter-policy]	[select_1_start]	[select_1_start]
3	[select_1_start]	[select_1_start]	[<acl-number>]	[select_1_start, <acl-number>]
4	[select_1_start]	[select_1_start, <acl-number>, |,]	[ip-prefix]	[select_1_start, <acl-number>, |, ip-prefix]
5	[ip-prefix]	[select_1_start, <acl-number>, |, ip-prefix]	[<ip-prefix-name>]	[select_1_start, <acl-number>, |, <ip-prefix-name>]
6	[select_1_start]	[select_1_start, <acl-number>, |, <ip-prefix-name>, |]	[acl-name]	[select_1_start, <acl-number>, |, <ip-prefix-name>, |, acl-name]
7	[acl-name]	[select_1_start, <acl-number>, #, <ip-prefix-name>, #, acl-name]	[<acl-name>]	[select_1_start, <acl-number>, |, <ip-prefix-name>, |, <acl-name>]
8	[<acl-number>, <ip-prefix-name>, <acl-name>]	[select_1_start, <acl-number>, |, <ip-prefix-name>, |, <acl-name>]	[select_1_end]	[select_1_end]

图 15:CLI 图构造的一个例子。

CLI 图模型构建:为了构建 CGM，我们在正式的语法验证中利用语法解析器

pyparsing 支持的 parse action 函数帮助构建图模型，它允许我们指定在成功匹配标记后要调用的函数。因此，我们定义了以下解析动作函数:

```
def leaf_gen(令牌):
    return [[" leaf ", {" name ", tokens [ 0]}]]
def select_gen(令牌):
    return [[" select ", tokens [0]。as List ()]]
def option_gen(令牌):
    return [[" option ", token[0]。as List ()]]
def ele_gen(令牌):
    如果代币。as List()！= []:
        返回[[" ele ", 代币。as List ()]]
    返还代币
```

语法分析器如图所示 5 通过上述解析操作，可以将 CLI 模板的平面字符串转换为嵌套结构，称为*clistruc*。以上面的 filter-policy 命令为例，它可以转换成如图所示的结构 16.

```
['ele', [
 ['leaf', {'name': 'filter-policy'}],
 ['select',[
  '(',
  ['ele', [['leaf', {'name': '<acl-number>'}]]],
  '|',
  ['ele',[['leaf', {'name': 'ip-prefix'}],['leaf', {'name': '<ip-prefix-name>'}]]],
  '/',
  ['ele',[['leaf', {'name': 'acl-name'}], ['leaf', {'name': '<acl-name>'}]]],
  ')']],
 ['select',[
  '(',
  ['ele', [['leaf', {'name': 'import'}]]],
  '|',
  ['ele', [['leaf', {'name': 'export'}]]],
  ')']]
]]
```

图 16:嵌套 CLI 结构的示例。

算法 2: CLI 图模型构建

```
1 Func get_syntax_graph(clistruc、cligraph、dict):Func
2     foreach ele clistruc岛
3 如果 is_symbol_or_leaf(ele)则
四          _ 符号_叶_过程(ele，cligraph，字典)
5 其他
6 字典信息=记录信息(字典)dict_ori
七          get_syntax_graph(ele[1]，cligraph，字典)
8 后处理(字典，字典)
          _
    _
```

以*clistruc*为输入，我们开发了一个递归算法来构造如图所示的 CLI 图 6. 递归 CGM 构造算法的框架如算法所示 2. CGM 的节点
cligraph 来自叶子或者像嵌套 clistruc 中的括号和大括号这样的符号。因此，我们在线上执行图形节点/边插入 4. 对于嵌套的 ele，我们递归地调用 get_syntax_graph(行 7) 来处理内部元素。然而，我们记录/处理
字典中的辅助信息(行 6 和 8) 重新草书呼叫之前/之后。结构 dict 主要维护 prev_stack 和 tail_stack，辅助节点/边插入。对于每个节点，我们添加从 prev_stack 中的节点到当前节点的有向边。存储可能具有更多子节点的已处理节点

算法 CLI 图模型 Con-的功能细节

结构

```
1 Func symbol_leaf_process(ele, cligraph, dict):Func
2 如果是分离符号(ele ),则
3 dict['tail_stack']。追加(' # ')
4 返回
5 如果 is_start_symbol(ele)则
6 节点= '_ '。join(字典['标签'][ele], len(字典['符号']), '开始')
7 dict['sym_stack']。追加(节点)
8 如果是结束符号(ele ),则
9 start_node = dict['sym_stack']。流行()
10 node = start _ node . replace(' start ', ' end ')
11 如果 is_option(node)则
12 cligraph.add_edge(起始节点, 节点)
13 如果 is_leaf(ele)则
14 节点= ele
15 cligraph.添加节点(节点)
16 foreach prev dict[' prev _ stack ']do
17 cligraph.add_edge(上一个, 节点)
18 dict['prev_stack']。清除()
19 dict['prev_stack']。追加(节点)
20 如果 is _ tail _ replace(dict[' tail _ stack '])则
21 字典['tail_stack'][-1] =节点
其他 22 个
23 字典['tail_stack']。追加(节点)
24 如果 is_end_symbol(ele)则
25 reshape_tail_stack(字典)
26 Func shape _ tail _ stack(dict):
27 tail _ stack = dict[' tail _ stack ']
28 结束节点=尾部堆栈[-1]
29 start _ node = end _ node . replace(' end ', ' start ')
30 start_ind = tail_stack.index(开始节点)
31 tail _ new = tail _ stack[0:start _ ind]+[end _ node]
32 dict['tail_stack'] = tail_new
33 Func record_infos(字典):
34 返回副本(字典)
35 Func 后处理(字典、字典、ele、clistruc):
36 下一个 ele = clistruc (ele_idx + 1)
37 prev _ stack _ ori = dict _ ori[' prev _ stack ']
38 如果 is_sep_symbol(next_ele)则
39 dict[' prev _ stack ']= copy(prev _ stack _ or)
其他 40 个
41 tail _ stack = dict[' tail _ stack ']
42 last_prev = prev_stack_ori[-1]
43 if last_prev tail_stack 然后
44 ind = tail _ stack . index(last _ prev)+1
45 dict[' prev _ stack ']= tail _ stack[ind:]
```

在 tail_stack。当退出对 ele 的递归调用时，我们可以根据 dict_ori 和 dict 中的两个堆栈来确定下一个 prev_stack。数字 15 演示前八个节点/边的插入

在图形中构造 CLI 图 6. 算法中显示了详细的 CGM 算法 2 和 3.

算法 CLI 实例-模板匹配的功能细节

```
1 Func match_next(next, next_candis, cligraph):
2 匹配标志=假
3 match_states = []
4 foreach candi next _ candis do
5 如果 candi 不存在，那么
6 继续
七 如果 is_keyword(cligraph,candi)和 next == candi 那么
8 匹配标志=真
9 match_states.append
10 if match_states! = []那么
11 返回{'match_flag': match_flag, ' match_states': match_states}
12 foreach candi next _ candis do
13 如果坎迪没有，那么
14 继续
15 如果 is_para(cligraph,candi)和 is_type_fit(下一个, candi)
然后
16 匹配标志=真
17 匹配_状态.追加(candi)
18 return {'match_flag': False, ' match_states': match_states}
19 Func get_next_candis(匹配状态, cligraph):
20 下一个州= []
21 个 foreach 项目匹配状态 do
22 cligraph.继任者(项目)
23 如果 succs == []，则
24 next _ States . extend([无])
25 每一次成功
26 next _ States . extend(get _ valid _ succs(succ, cligraph, []))
27 返回集合(next_states)
28 Func get_valid_succssors(节点, cligraph, 已访问):
29 如果 is_valid_node(cligraph, 节点)那么
30 返回[节点]
31 如果cligraph.后继者(节点)== []那么
32 返回[无]
33 valid_succ = []
34 cligraph.的每一次成功继任者(节点)做什么
35 如果∉成功访问了
36 visited.append(succ)
37 valid _ succ . extend(get _ valid _ succssors(succ, cligraph, 参观过))
```

D NETBERT 的详细评估

我们展示了 Mapper 映射性能的详细评估结果，此外 7.3. 我们增加了另一个指标:平均循环排名(MRR)。推荐参数列表的倒数排名是第一个正确答案排名的倒数。平均倒数排名是所有测试用例的倒数排名的平均值。较高的 MRR 意味着建议的 Mapper 参数更准确。

映射设置	模型	召回中的k @前k (%)												维护、修理和更换
		一	2	3	四	5	6	七	8	9	10	20	30	
华为-UDM	红外辐射（InfraredRadiatio	41	52	61	66	69	74	76	78	79	80	90	93	0.5401

表6:映射器性能