使用 NVIDIA 的对话 AI

NetApp Solutions

NetApp
September 10, 2024

This PDF was generated from https://docs.netapp.com/zh-cn/netapp-solutions/ai/cainvidia_solution_overview.html on September 10, 2024. Always check docs.netapp.com for the latest.

目录

WP-7328： 使用 NVIDIA JarVis 的 NetApp 对话 AI

Rick Huang， Sung-Han Lin， NetApp， NVIDIA 公司的 Dide Onofino

NVIDIA DGX 系统系列由全球首款基于人工智能（AI）的集成系统组成，这些系统专为企业 AI 而构建。NetApp AFF 存储系统可提供极致性能和行业领先的混合云数据管理功能。NetApp 和 NVIDIA 合作创建了 NetApp ONTAP AI 参考架构，这是一款适用于人工智能和机器学习（ML）的统包解决方案工作负载，可提供企业级性能，可靠性和支持。

本白皮书为客户构建对话式 AI 系统提供了方向性指导，以支持各个行业的不同使用情形。其中包括有关使用 NVIDIA JarVis 部署系统的信息。这些测试是使用 NVIDIA DGX 工作站和 NetApp AFF A220 存储系统执行的。

解决方案的目标受众包括以下组：

- 企业架构师，负责设计用于开发人工智能模型和软件的解决方案，用于虚拟零售助理等人工智能对话用例
- 寻求高效方式实现语言建模开发目标的数据科学家
- 负责维护和处理客户问题和对话记录等文本数据的数据工程师
- 有兴趣转变对话式 AI 体验并加快 AI 计划上市速度的高管和 IT 决策者以及业务主管

解决方案概述

本文档介绍了适用于ONTAP AI和NVIDIA DGX的对话式AI模型。

NetApp ONTAP AI和BlueXP复制和同步

由 NVIDIA DGX 系统和 NetApp 云连接存储系统提供支持的 NetApp ONTAP AI 架构由 NetApp 和 NVIDIA 开发并验证。此参考架构为 IT 组织提供了以下优势：

- 消除设计复杂性
- 支持独立扩展计算和存储
- 支持客户从小规模入手，无缝扩展
- 为各种性能和成本点提供一系列存储选项 NetApp ONTAP AI 可将 DGX 系统和 NetApp AFF A220 存储系统与一流的网络紧密集成在一起。NetApp ONTAP AI 和 DGX 系统消除了设计复杂性和猜测性工作，从而简化了 AI 部署。客户可以从小规模入手，无中断地扩展系统，同时智能地管理从边缘到核心再到云再到云的数据。

借助NetApp BlueXP复制和同步功能、您可以通过各种协议轻松移动数据、无论是在两个NFS共享、两个CIFS共享之间、还是在一个文件共享与Amazon S3、Amazon Elelic File System (EFS)或Azure Blb存储之间。主动 - 主动操作意味着您可以继续同时使用源和目标，并在需要时逐步同步数据更改。BlueXP复制和同步功能支持您在任何源系统和目标系统(无论是内部系统还是基于云的系统)之间移动和增量同步数据、为您使用数据开辟了多种新方式。在内部系统，云入网和云迁移或协作和数据分析之间迁移数据变得非常容易。下图显示了可用的源和目标。

在对话式AI系统中、开发人员可以利用BlueXP Copy and Sync将对话历史记录从云归档到数据中心、以便对自然语言处理(NLP)模型进行离线训练。通过培训模式识别更多意向，对话式 AI 系统将更好地处理最终用户提出的更复杂的问题。

NVIDIA JarVis 多模式框架

"NVIDIA JarVis" 是一个端到端框架，用于构建对话式 AI 服务。它包括以下经过 GPU 优化的服务：

- 自动语音识别（Automatic Speech Recognition， As1）
- 自然语言理解（NLF）
- 与域特定的履行服务集成
- 文本语音转换（TTS- 语音转换）
- 基于计算机视觉（CV） Jarvis 的服务使用最先进的深度学习模型来应对实时对话 AI 这一复杂且极具挑战性的任务。要与最终用户进行实时自然的交互，模型需要在 300 毫秒内完成计算。自然交互具有挑战性，需要多模式感知集成。模型管道也很复杂，需要在上述服务之间进行协调。

JarVis 是一个完全加速的应用程序框架，用于构建使用端到端深度学习管道的多模式对话 AI 服务。JARVIS 框架包括经过预先培训的人工智能对话模型，工具以及针对语音，视觉和 NLU 任务优化的端到端服务。除了 AI 服务之外， JarVis 还支持您同时融合视觉，音频和其他传感器输入，以便在虚拟助手，多用户化和呼叫中心助理等应用程序中提供多用户，多上下文对话等功能。

NVIDIA Nemo

"NVIDIA Nemo" 是一款开源 Python 工具包，用于使用易于使用的应用程序编程接口（API）构建，培训和微调 GPU 加速的一流对话 AI 模型。Nemo 使用 NVIDIA GPU 中的 Tensor 核心运行混合精度计算，并可轻松扩展到多个 GPU ，以提供尽可能高的训练性能。Nemo 用于为视频呼叫记录，智能视频助理以及医疗保健，金融，零售和电信等不同行业的自动呼叫中心支持等实时应用程序构建模型。

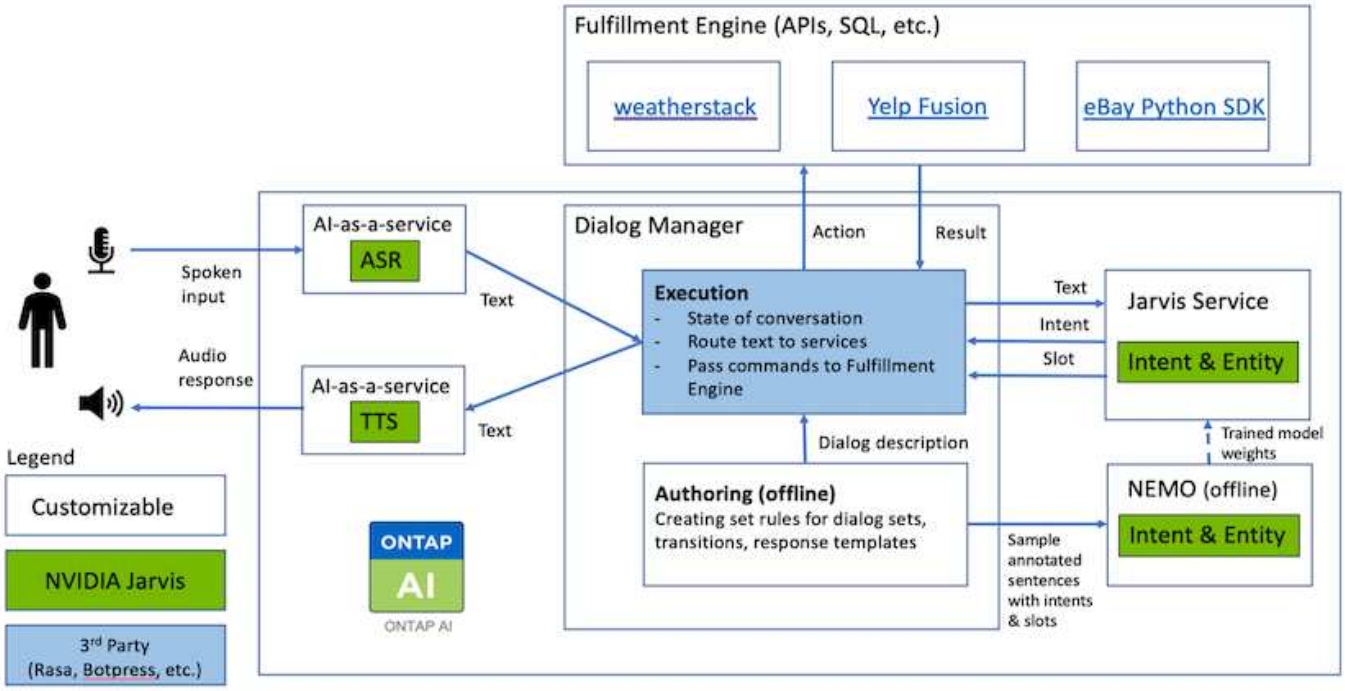
我们使用 Nemo 来训练模型，以便识别归档对话历史记录中用户问题的复杂意图。此培训将零售虚拟助手的功能扩展到了 Jarvis 所提供的功能之外。

零售用例摘要

我们使用 NVIDIA Jarvis 构建了一个虚拟零售助理，可接受语音或文本输入并回答有关天气，关注点和库存定价的问题。对话式 AI 系统能够记住对话流，例如，如果用户未指定天气或感兴趣点的位置，可以询问跟进问题。系统还可以识别诸如 " 泰国食品 " 或 " 笔记本电脑内存 " 等复杂实体。 它了解自然语言问题，例如 " 下星期在洛杉矶会下雨吗？ " 有关零售虚拟助手的演示，请参见 "针对零售用例自定义状态和流程"。

解决方案技术

下图显示了建议的对话 AI 系统架构。您可以使用语音信号或文本输入与系统进行交互。如果检测到语音输入，则 JARVIS AI as-service （AIaaS）将执行 As1 ，以便为对话框管理器生成文本。对话框管理器会记住对话状态，将文本路由到相应的服务，并将命令传递到实施引擎。JARVIS NLP 服务会输入文本，识别意向和实体，并将这些意向和实体插槽输出回对话框管理器，然后由该对话框管理器向执行引擎发送操作。履行引擎由问题解答用户查询的第三方 API 或 SQL 数据库组成。从实施引擎收到结果后，对话管理器会将文本路由到 JarVis TTSAIaaS ，以便为最终用户生成音频响应。我们可以归档对话历史记录，为 Nemo 培训添加意向和插槽，以便随着更多用户与系统交互， NLP 服务得到改进。



硬件要求

此解决方案已通过一个 DGX 工作站和一个 AFF A220 存储系统的验证。JARVIS 需要使用 T4 或 V100 GPU 来执行深度神经网络计算。

下表列出了在测试中实施解决方案所需的硬件组件。

硬件	数量
T4 或 V100 GPU	1.

硬件	数量
NVIDIA DGX Station	1.

软件要求

下表列出了在测试中实施解决方案所需的软件组件。

软件	版本或其他信息
NetApp ONTAP 数据管理软件	9.6
Cisco NX-OS 交换机固件	7.0 （3） I6 （1）
NVIDIA DGX 操作系统	4.0.4 — Ubuntu 18.04 LTS
NVIDIA JarVis Framework	EA v0.2
NVIDIA Nemo	nvcr.io/nvidia/nemo： v0.10
Docker 容器平台	18.06.1-ce [e68fc7a]

概述

本节详细介绍了虚拟零售助理的实施。

JarVis 部署

您可以注册 "JARVIS 早期访问计划" 访问 NVIDIA GPU Cloud （NGC）上的 JarVis 容器。从 NVIDIA 收到凭据后，您可以使用以下步骤部署 JarVis：

1. 登录到 NGC 。
2. 在 NGC 上设置您的组织： `ea-2-JarVis` 。
3. 找到 JarVis EA v0.2 资产： JarVis containers are in `Private Registry` > `Organization Containers` 。
4. 选择 JarVis ：导航到 `Model Scripts` ，然后单击 `JarVis Quick Start`
5. 验证所有资产是否均正常工作。
6. 查找用于构建您自己的应用程序的文档： PDF 可在 `Model Scripts` > `JarVis Documentation` > `File Browser` 中找到。

针对零售用例自定义状态和流程

您可以针对特定使用情形自定义对话框管理器的状态和流。在我们的零售示例中，我们提供了以下四个 YAML 文件，用于根据不同意向指导对话。

查看以下文件名列表以及每个文件的问题描述：

- `main_flow.yml` ：定义主要对话流和状态，并在必要时将此流定向到其他三个 YAML 文件。

- `retail _flow.yml`： 包含与零售或感兴趣点问题相关的状态。系统会提供最近商店的信息或给定商品的价格。
- `weather flow.yml`： 包含与天气问题相关的状态。如果无法确定位置，系统会询问一个跟进问题以进行澄清。
- `error_flow.yml`： 处理用户意向不属于上述三个 YAML 文件的情况。显示错误消息后，系统会重新路由到接受用户问题。以下各节包含这些 YAML 文件的详细定义。

main_flow.yml

```
name: JarvisRetail
intent_transitions:
  jarvis_error: error
  price_check: retail_price_check
  inventory_check: retail_inventory_check
  store_location: retail_store_location
  weather.weather: weather
  weather.temperature: temperature
  weather.sunny: sunny
  weather.cloudy: cloudy
  weather.snow: snow
  weather.rainfall: rain
  weather.snow_yes_no: snowfall
  weather.rainfall_yes_no: rainfall
  weather.temperature_yes_no: tempyesno
  weather.humidity: humidity
  weather.humidity_yes_no: humidity
  navigation.startnavigationpoi: retail # Transitions should be context
and slot based. Redirecting for now.
  navigation.geteta: retail
  navigation.showdirection: retail
  navigation.showmappoi: idk_what_you_talkin_about
  nomatch.none: idk_what_you_talkin_about
states:
  init:
    type: message_text
    properties:
      text: "Hi, welcome to NARA retail and weather service. How can I
help you?"
  input_intent:
    type: input_context
    properties:
      nlp_type: jarvis
      entities:
        intent: dontcare
# This state is executed if the intent was not understood
  dont_get_the intent:
```

```
    type: message_text_random
    properties:
      responses:
        - "Sorry I didn't get that! Please come again."
        - "I beg your pardon! Say that again?"
        - "Are we talking about weather? What would you like to know?"
        - "Sorry I know only about the weather"
        - "You can ask me about the weather, the rainfall, the
temperature, I don't know much more"
      delay: 0
    transitions:
      next_state: input_intent
  idk_what_you_talkin_about:
    type: message_text_random
    properties:
      responses:
        - "Sorry I didn't get that! Please come again."
        - "I beg your pardon! Say that again?"
        - "Are we talking about retail or weather? What would you like to
know?"
        - "Sorry I know only about retail and the weather"
        - "You can ask me about retail information or the weather, the
rainfall, the temperature. I don't know much more."
      delay: 0
    transitions:
      next_state: input_intent
  error:
    type: change_context
    properties:
        update_keys:
           intent: 'error'
    transitions:
        flow: error_flow
  retail_inventory_check:
    type: change_context
    properties:
        update_keys:
           intent: 'retail_inventory_check'
    transitions:
        flow: retail_flow
  retail_price_check:
    type: change_context
    properties:
        update_keys:
           intent: 'check_item_price'
    transitions:
```

```
        flow: retail_flow
  retail_store_location:
    type: change_context
    properties:
        update_keys:
           intent: 'find_the_store'
    transitions:
        flow: retail_flow
  weather:
    type: change_context
    properties:
        update_keys:
           intent: 'weather'
    transitions:
        flow: weather_flow
  temperature:
    type: change_context
    properties:
        update_keys:
           intent: 'temperature'
    transitions:
        flow: weather_flow
  rainfall:
    type: change_context
    properties:
        update_keys:
           intent: 'rainfall'
    transitions:
        flow: weather_flow
  sunny:
    type: change_context
    properties:
        update_keys:
           intent: 'sunny'
    transitions:
        flow: weather_flow
  cloudy:
    type: change_context
    properties:
        update_keys:
           intent: 'cloudy'
    transitions:
        flow: weather_flow
  snow:
    type: change_context
    properties:
```

```
        update_keys:
           intent: 'snow'
    transitions:
        flow: weather_flow
  rain:
    type: change_context
    properties:
        update_keys:
           intent: 'rain'
    transitions:
        flow: weather_flow
  snowfall:
      type: change_context
      properties:
          update_keys:
             intent: 'snowfall'
      transitions:
          flow: weather_flow
  tempyesno:
      type: change_context
      properties:
          update_keys:
             intent: 'tempyesno'
      transitions:
          flow: weather_flow
  humidity:
      type: change_context
      properties:
          update_keys:
             intent: 'humidity'
      transitions:
          flow: weather_flow
  end_state:
    type: reset
    transitions:
      next_state: init
```

Retail， flow.yml

```
name: retail_flow
states:
  store_location:
    type: conditional_exists
    properties:
      key: '{{location}}'
```

```
    transitions:
      exists: retail_state
      notexists: ask_retail_location
  retail_state:
    type: Retail
    properties:
    transitions:
      next_state: output_retail
  output_retail:
      type: message_text
      properties:
        text: '{{retail_status}}'
      transitions:
        next_state: input_intent
  ask_retail_location:
    type: message_text
    properties:
      text: "For which location? I can find the closest store near you."
    transitions:
      next_state: input_retail_location
  input_retail_location:
    type: input_user
    properties:
      nlp_type: jarvis
      entities:
        slot: location
      require_match: true
    transitions:
      match: retail_state
      notmatch: check_retail_jarvis_error
  output_retail_acknowledge:
    type: message_text_random
    properties:
      responses:
        - 'ok in {{location}}'
        - 'the store in {{location}}'
        - 'I always wanted to shop in {{location}}'
      delay: 0
    transitions:
      next_state: retail_state
  output_retail_notlocation:
    type: message_text
    properties:
      text: "I did not understand the location. Can you please repeat?"
    transitions:
      next_state: input_intent
```

```
  check_rerail_jarvis_error:
    type: conditional_exists
    properties:
      key: '{{jarvis_error}}'
    transitions:
      exists: show_retail_jarvis_api_error
      notexists: output_retail_notlocation
  show_retail_jarvis_api_error:
    type: message_text
    properties:
      text: "I am having troubled understanding right now. Come again on
that?"
    transitions:
      next_state: input_intent
```

weather flow.yml

```
name: weather_flow
states:
  check_weather_location:
    type: conditional_exists
    properties:
      key: '{{location}}'
    transitions:
      exists: weather_state
      notexists: ask_weather_location
  weather_state:
    type: Weather
    properties:
    transitions:
      next_state: output_weather
  output_weather:
      type: message_text
      properties:
        text: '{{weather_status}}'
      transitions:
        next_state: input_intent
  ask_weather_location:
    type: message_text
    properties:
      text: "For which location?"
    transitions:
      next_state: input_weather_location
  input_weather_location:
    type: input_user
```

```
    properties:
      nlp_type: jarvis
      entities:
        slot: location
      require_match: true
    transitions:
      match: weather_state
      notmatch: check_jarvis_error
  output_weather_acknowledge:
    type: message_text_random
    properties:
      responses:
        - 'ok in {{location}}'
        - 'the weather in {{location}}'
        - 'I always wanted to go in {{location}}'
      delay: 0
    transitions:
      next_state: weather_state
  output_weather_notlocation:
    type: message_text
    properties:
      text: "I did not understand the location, can you please repeat?"
    transitions:
      next_state: input_intent
  check_jarvis_error:
    type: conditional_exists
    properties:
      key: '{{jarvis_error}}'
    transitions:
      exists: show_jarvis_api_error
      notexists: output_weather_notlocation
  show_jarvis_api_error:
    type: message_text
    properties:
      text: "I am having troubled understanding right now. Come again on
that, else check jarvis services?"
    transitions:
      next_state: input_intent
```

error_flow.yml

```
name: error_flow
states:
  error_state:
    type: message_text_random
    properties:
      responses:
        - "Sorry I didn't get that!"
        - "Are we talking about retail or weather? What would you like to
know?"
        - "Sorry I know only about retail information or the weather"
        - "You can ask me about retail information or the weather, the
rainfall, the temperature. I don't know much more"
        - "Let's talk about retail or the weather!"
      delay: 0
    transitions:
      next_state: input_intent
```

以履行引擎的形式连接到第三方 API

我们将以下第三方 API 作为履行引擎连接到问题解答问题：

- "WeatherStack API"：返回给定位置的天气，温度，雨和雪。
- "Yelp Fusion API"：返回给定位置中最接近的存储信息。
- "eBay Python SDK"：返回给定项目的价格。

NetApp 零售助理演示

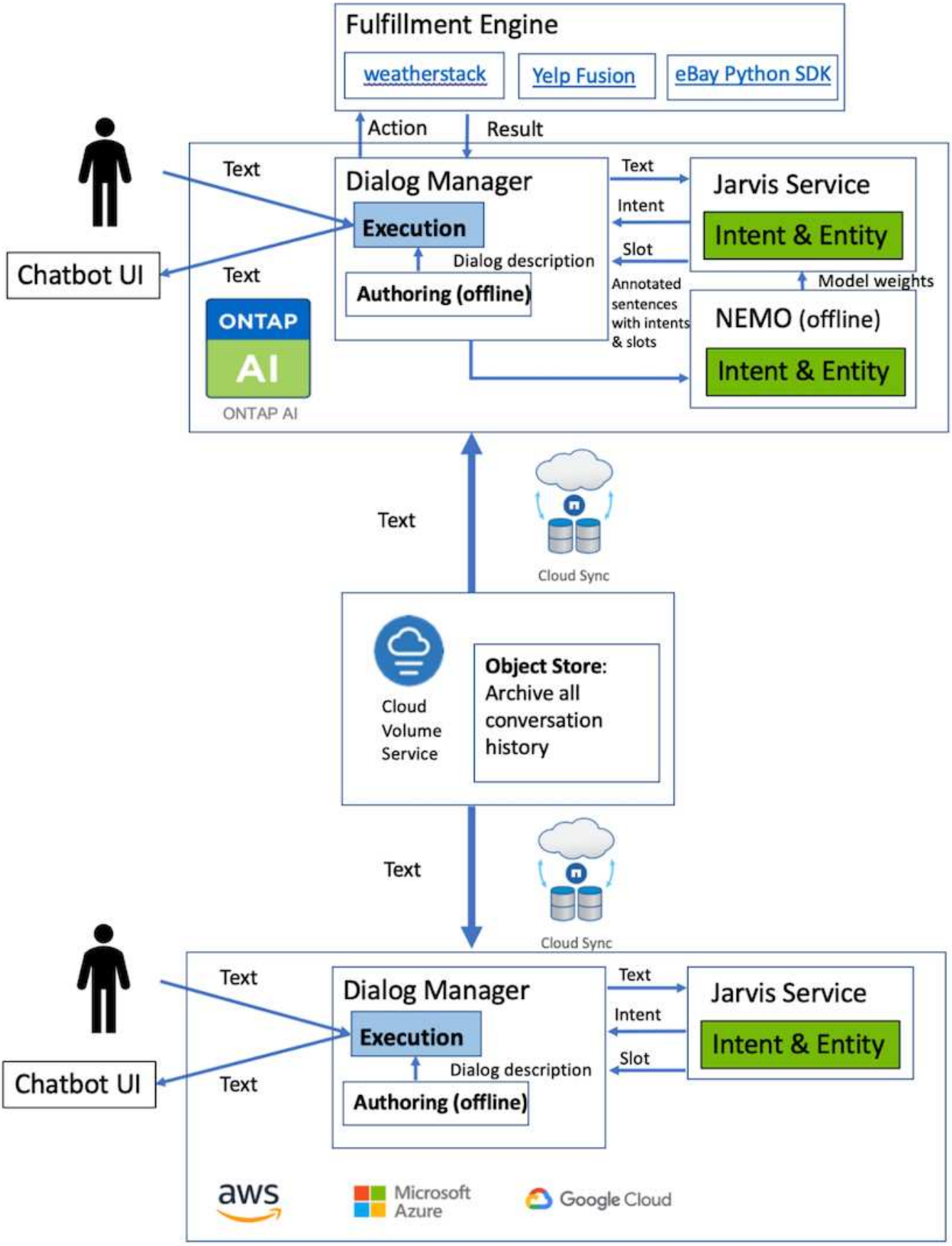
我们录制了 NetApp 零售助理（ Nara ）的演示视频。

视频演示的一个例子

视频演示的一个例子

使用**NetApp BlueXP**复制和同步以存档对话历史记录

通过每天将对话历史记录转储到CSV文件一次、我们可以利用BlueXP Copy and Sync将日志文件下载到本地存储。下图显示了在内部和公有云中部署Jarvis、同时使用BlueXP Copy and Sync为Nemo培训发送对话历史记录的架构。有关 Nemo 培训的详细信息，请参见一节 "使用 Nemo 培训扩展意向模型"。


Fulfillment Engine
weatherstack
Yelp Fusion
eBay Python SDK
Action
Result
Text
Chatbot UI
Text
ONTAP
AI
ONTAP AI
Dialog Manager
Execution
Dialog description
Authoring (offline)
Text
Intent
Slot
Annotated sentences with intents & slots
Jarvis Service
Intent & Entity
Model weights
NEMO (offline)
Intent & Entity
Text
Cloud Sync
Cloud Volume Service
Object Store: Archive all conversation history
Text
Cloud Sync
Text
Chatbot UI
Text
Dialog Manager
Execution
Dialog description
Authoring (offline)
Text
Intent
Slot
Jarvis Service
Intent & Entity
aws
Microsoft Azure
Google Cloud

使用 **Nemo** 培训扩展意向模型

NVIDIA Nemo 是由 NVIDIA 构建的一个工具包，用于创建对话式 AI 应用程序。该工具包包含一系列针对 ASL ， NLP 和 TTS- 的预培训模块，使研究人员和数据科学家能够轻松构建复杂的神经网络架构，并更加专注于设计自己的应用程序。

如上例所示， Nara 只能处理有限类型的问题。这是因为经过预先培训的 NLP 模型只会对这些类型的问题进行训练。如果我们希望 Nara 能够处理更广泛的问题，我们需要使用自己的数据集对其进行重新训练。因此，我们将在此演示如何使用 Nemo 扩展 NLP 模型以满足要求。我们首先将从 Nara 收集的日志转换为 Nemo 的格式，然后训练数据集以增强 NLP 模型。

型号

我们的目标是使 Nara 能够根据用户首选项对项目进行排序。例如，我们可能会要求 Nara 推荐排名最高的寿司店，也可能希望 Nara 寻找价格最低的 jeans 。为此，我们使用 Nemo 中提供的意向检测和插槽填充模型作为我们的培训模型。通过此模型， Nara 可以了解搜索首选项的意图。

数据准备

为了训练模型，我们会收集此类问题的数据集，并将其转换为 Nemo 格式。我们在此处列出了用于训练模型的文件。

dict.intents.csv

此文件列出了我们希望 Nemo 了解的所有意向。此处，我们有两个主要意向，一个意图仅用于对不符合任何主要意向的问题进行分类。

```
price_check
find_the_store
unknown
```

dict.slots.csv

此文件列出了我们可以在培训问题上标记的所有插槽。

```
B-store.type
B-store.name
B-store.status
B-store.hour.start
B-store.hour.end
B-store.hour.day
B-item.type
B-item.name
B-item.color
B-item.size
B-item.quantity
B-location
B-cost.high
```

```
B-cost.average
B-cost.low
B-time.period_of_time
B-rating.high
B-rating.average
B-rating.low
B-interrogative.location
B-interrogative.manner
B-interrogative.time
B-interrogative.personal
B-interrogative
B-verb
B-article
I-store.type
I-store.name
I-store.status
I-store.hour.start
I-store.hour.end
I-store.hour.day
I-item.type
I-item.name
I-item.color
I-item.size
I-item.quantity
I-location
I-cost.high
I-cost.average
I-cost.low
I-time.period_of_time
I-rating.high
I-rating.average
I-rating.low
I-interrogative.location
I-interrogative.manner
I-interrogative.time
I-interrogative.personal
I-interrogative
I-verb
I-article
O
```

Traine.tsv

这是主要的培训数据集。每行都以文件 dict.intent.csv 中列出的意图类别后面的问题开头。此标签将从零开始枚举。

Train_slots.tsv

```
20 46 24 25 6 32 6
52 52 24 6
23 52 14 40 52 25 6 32 6
…
```

训练模型

```
docker pull nvcr.io/nvidia/nemo:v0.10
```

然后，我们将使用以下命令启动此容器。在此命令中，我们会将容器限制为使用单个 GPU （ GPU ID = 1 ），因为这是一项轻型训练练习。此外，我们还会将本地工作空间 /workstore/nemo/ 映射到容器 /nemo 中的文件夹。

```
NV_GPU='1' docker run --runtime=nvidia -it --shm-size=16g \
                        --network=host --ulimit memlock=-1 --ulimit
stack=67108864 \
                        -v /workspace/nemo:/nemo\
                        --rm nvcr.io/nvidia/nemo:v0.10
```

在容器中，如果要从最初的预先培训的 Bert 模型开始，我们可以使用以下命令启动培训操作步骤。data_dir 是用于设置训练数据路径的参数。work_dir 用于配置检查点文件的存储位置。

```
cd examples/nlp/intent_detection_slot_tagging/
python joint_intent_slot_with_bert.py \
    --data_dir /nemo/training_data\
    --work_dir /nemo/log
```

如果我们有新的培训数据集并希望改进先前的模型，则可以使用以下命令从停止的位置继续操作。checkpoint_dir 获取上一个检查点文件夹的路径。

```
cd examples/nlp/intent_detection_slot_tagging/
python joint_intent_slot_infer.py \
    --data_dir /nemo/training_data \
    --checkpoint_dir /nemo/log/2020-05-04_18-34-20/checkpoints/ \
    --eval_file_prefix test
```

推理模型

我们需要在经过一定次数的时间之后验证经过训练的模型的性能。使用以下命令，我们可以逐个测试查询。例如，在此命令中，我们希望检查我们的模型是否能够正确识别查询的目的 在哪里可以获得最好的意大利面 。

```
cd examples/nlp/intent_detection_slot_tagging/
python joint_intent_slot_infer_b1.py \
--checkpoint_dir /nemo/log/2020-05-29_23-50-58/checkpoints/ \
--query "where can i get the best pasta" \
--data_dir /nemo/training_data/ \
--num_epochs=50
```

然后，以下是推理的输出。在输出中，我们可以看到经过培训的模型可以正确预测 DETAINT_the_store 的意向，并返回我们感兴趣的关键字。通过这些关键字，我们可以使 Nara 搜索用户所需内容并进行更精确的搜索。

```
[NeMo I 2020-05-30 00:06:54 actions:728] Evaluating batch 0 out of 1
[NeMo I 2020-05-30 00:06:55 inference_utils:34] Query: where can i get the
best pasta
[NeMo I 2020-05-30 00:06:55 inference_utils:36] Predicted intent:       1
find_the_store
[NeMo I 2020-05-30 00:06:55 inference_utils:50] where   B-
interrogative.location
[NeMo I 2020-05-30 00:06:55 inference_utils:50] can     O
[NeMo I 2020-05-30 00:06:55 inference_utils:50] i       O
[NeMo I 2020-05-30 00:06:55 inference_utils:50] get     B-verb
[NeMo I 2020-05-30 00:06:55 inference_utils:50] the     B-article
[NeMo I 2020-05-30 00:06:55 inference_utils:50] best    B-rating.high
[NeMo I 2020-05-30 00:06:55 inference_utils:50] pasta   B-item.type
```

结论

一个真正的人工智能对话系统可以进行类似于人类的对话，了解相关背景并提供智能响应。此类 AI 模型通常非常庞大且非常复杂。借助 NVIDIA GPU 和 NetApp 存储，可以对大规模的一流语言模型进行培训和优化，以快速运行推理。这是一个重大的进步，旨在结束快速 AI 模型与大型复杂 AI 模型之间的权衡。GPU 优化的语言理解模式可以集成到医疗保健，零售和金融服务等行业的 AI 应用程序中，为智能扬声器和客户服务线中的高级数字语音助理提供支持。通过这些高质量的对话式 AI 系统，各个垂直行业的企业可以在与客户接洽时提供以前无法实现的个性化服务。

借助 JARVIS ，可以部署虚拟助手，数字 avatars ，多模式传感器 Fusion （ CV 与 ASS/NLP/TTs 融合）或任何 ASS/NLP/TTS/CV 独立用例，如记录。我们构建了一个虚拟零售助理，可以对天气，关注点和库存定价方面的问题进行问题解答处理。我们还演示了如何通过使用BlueXP Copy and Sync归档对话历史记录并对Nemo模型进行新数据训练来提高对话AI系统的自然语言理解能力。

致谢

作者衷心感谢我们尊敬的 NVIDIA 同事：达维德·奥诺弗罗， Alex Qi ， Sicong Ji ， Marty Jain 和 Robert Sohigian 为本白皮书所做的贡献。作者还想感谢 NetApp 团队的主要

成员所做的贡献： Santosh Rao ， David Arnette ， Michael Oglesby ， Brent Davis ， Andy Sayare ， Erik Minder 和 Mike McNamara 。

我们衷心感谢所有这些人士，他们提供了洞察力和专业知识，为本白皮书的编写提供了极大的帮助。

从何处查找追加信息

要了解有关本文档中所述信息的更多信息，请参见以下资源：

- NVIDIA DGX Station ， V100 GPU ， GPU Cloud
  - NVIDIA DGX Stationhttps://www.nvidia.com/en-us/data-center/dgx-station/["https://www.nvidia.com/en-us/data-center/dgx-station/"^]
  - NVIDIA V100 Tensor 核心 GPUhttps://www.nvidia.com/en-us/data-center/tesla-v100/["https://www.nvidia.com/en-us/data-center/tesla-v100/"^]
  - NVIDIA NGChttps://www.nvidia.com/en-us/gpu-cloud/["https://www.nvidia.com/en-us/gpu-cloud/"^]
- NVIDIA JarVis 多模式框架
  - NVIDIA JarVishttps://developer.nvidia.com/nvidia-jarvis["https://developer.nvidia.com/nvidia-jarvis"^]
  - NVIDIA JarVis Early Accesshttps://developer.nvidia.com/nvidia-jarvis-early-access["https://developer.nvidia.com/nvidia-jarvis-early-access"^]
- NVIDIA Nemo
  - NVIDIA Nemohttps://developer.nvidia.com/nvidia-nemo["https://developer.nvidia.com/nvidia-nemo"^]
  - 开发人员指南https://nvidia.github.io/NeMo/["https://nvidia.github.io/NeMo/"^]
- NetApp AFF 系统
  - NetApp AFF A 系列产品规格https://www.netapp.com/us/media/ds-3582.pdf["https://www.netapp.com/us/media/ds-3582.pdf"^]
  - 适用于全闪存 FAS 的 NetApp 闪存优势https://www.netapp.com/us/media/ds-3733.pdf["https://www.netapp.com/us/media/ds-3733.pdf"^]
  - ONTAP 9 信息库http://mysupport.netapp.com/documentation/productlibrary/index.html?productID=62286["http://mysupport.netapp.com/documentation/productlibrary/index.html?productID=62286"^]
  - NetApp ONTAP FlexGroup Volumes 技术报告https://www.netapp.com/us/media/tr-4557.pdf["https://www.netapp.com/us/media/tr-4557.pdf"^]
- NetApp ONTAP AI
  - 采用 DGX-1 的 ONTAP AI 和 Cisco 网络设计指南https://www.netapp.com/us/media/nva-1121-design.pdf["https://www.netapp.com/us/media/nva-1121-design.pdf"^]
  - 《采用 DGX-1 的 ONTAP AI 和 Cisco 网络部署指南》https://www.netapp.com/us/media/nva-1121-deploy.pdf["https://www.netapp.com/us/media/nva-1121-deploy.pdf"^]
  - 采用 DGX-1 和 Mellanox 网络设计指南的 ONTAP AIhttp://www.netapp.com/us/media/nva-1138-design.pdf["http://www.netapp.com/us/media/nva-1138-design.pdf"^]
  - 采用 DGX-2 的 ONTAP AI 设计指南https://www.netapp.com/us/media/nva-1135-design.pdf["https://www.netapp.com/us/media/nva-1135-design.pdf"^]

版权信息

版权所有 © 2024 NetApp, Inc.。保留所有权利。中国印刷。未经版权所有者事先书面许可，本文档中受版权保护的任何部分不得以任何形式或通过任何手段（图片、电子或机械方式，包括影印、录音、录像或存储在电子检索系统中）进行复制。

从受版权保护的 NetApp 资料派生的软件受以下许可和免责声明的约束：

本软件由 NetApp 按“原样”提供，不含任何明示或暗示担保，包括但不限于适销性以及针对特定用途的适用性的隐含担保，特此声明不承担任何责任。在任何情况下，对于因使用本软件而以任何方式造成的任何直接性、间接性、偶然性、特殊性、惩罚性或后果性损失（包括但不限于购买替代商品或服务；使用、数据或利润方面的损失；或者业务中断），无论原因如何以及基于何种责任理论，无论出于合同、严格责任或侵权行为（包括疏忽或其他行为），NetApp 均不承担责任，即使已被告知存在上述损失的可能性。

NetApp 保留在不另行通知的情况下随时对本文档所述的任何产品进行更改的权利。除非 NetApp 以书面形式明确同意，否则 NetApp 不承担因使用本文档所述产品而产生的任何责任或义务。使用或购买本产品不表示获得 NetApp 的任何专利权、商标权或任何其他知识产权许可。

本手册中描述的产品可能受一项或多项美国专利、外国专利或正在申请的专利的保护。

有限权利说明：政府使用、复制或公开本文档受 DFARS 252.227-7013（2014 年 2 月）和 FAR 52.227-19（2007 年 12 月）中“技术数据权利 — 非商用”条款第 (b)(3) 条规定的限制条件的约束。

本文档中所含数据与商业产品和/或商业服务（定义见 FAR 2.101）相关，属于 NetApp, Inc. 的专有信息。根据本协议提供的所有 NetApp 技术数据和计算机软件具有商业性质，并完全由私人出资开发。 美国政府对这些数据的使用权具有非排他性、全球性、受限且不可撤销的许可，该许可既不可转让，也不可再许可，但仅限在与交付数据所依据的美国政府合同有关且受合同支持的情况下使用。除本文档规定的情形外，未经 NetApp, Inc. 事先书面批准，不得使用、披露、复制、修改、操作或显示这些数据。美国政府对国防部的授权仅限于 DFARS 的第 252.227-7015(b)（2014 年 2 月）条款中明确的权利。

商标信息

NetApp、NetApp 标识和 http://www.netapp.com/TM 上所列的商标是 NetApp, Inc. 的商标。其他公司和产品名称可能是其各自所有者的商标。