使用Jarvis、BlueXP Copy and Sync和Nemo构建虚拟助手

NetApp Solutions

NetApp
April 12, 2024

目录

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

```yaml
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

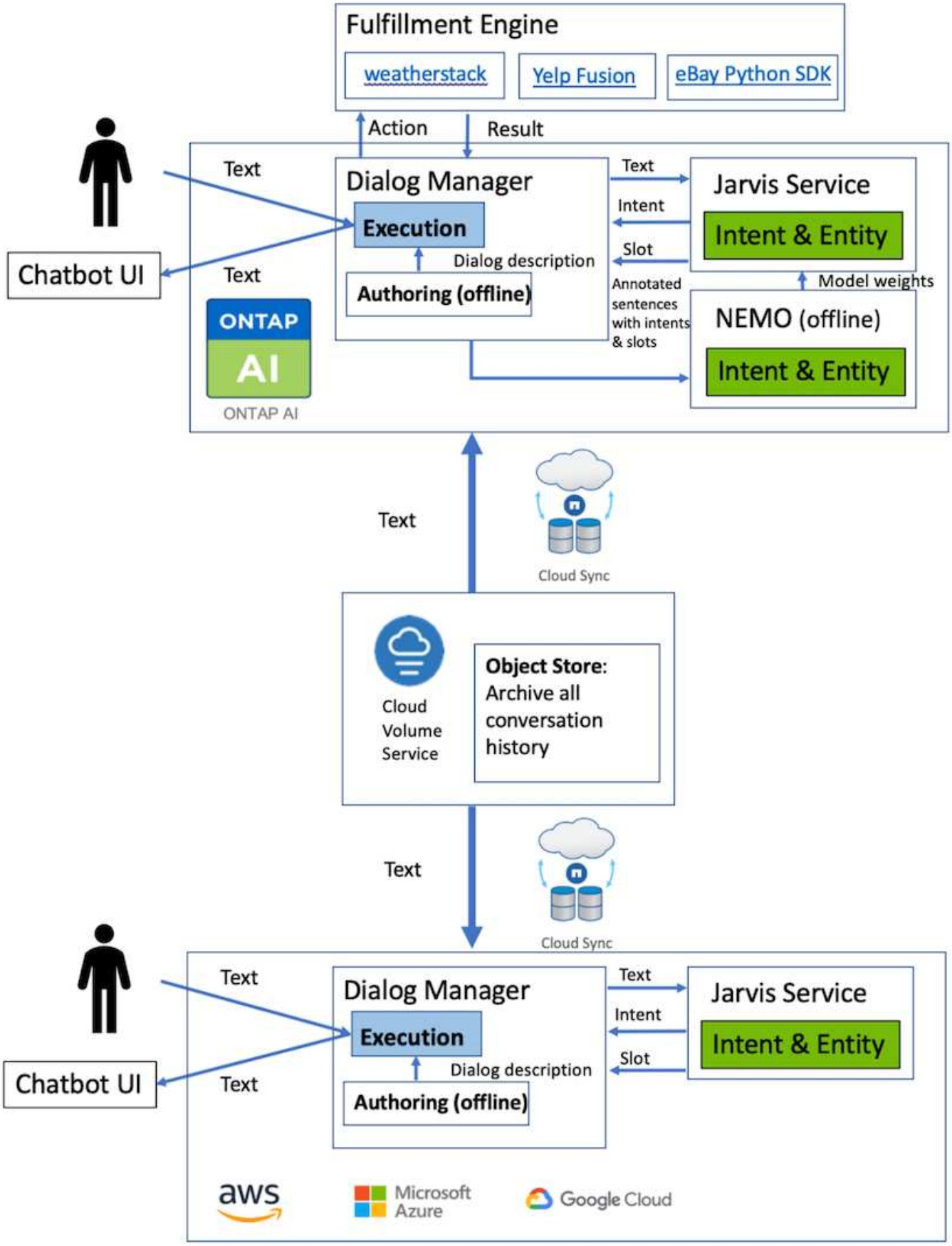
我们录制了 NetApp 零售助理（ Nara ）的演示视频。

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
```