使用 Jarvis 、 BlueXP Copy and Sync 和 Nemo 建立虛擬助理

NetApp Solutions

NetApp
April 12, 2024

目錄

總覽

本節詳細說明虛擬零售助理的實作方式。

JARVIS部署

您可以註冊 "JARVIS早鳥方案" 以存取NVIDIA GPU Cloud（NGC）上的JARVIS容器。從NVIDIA接收認證之後、您可以使用下列步驟來部署JARVIS：

1. 登入NGC。
2. 將您的組織設在NGC：「ea - 2-jarvis」。
3. 找到JARVIS EA v0.2資產：JARVIS Container位於「Private登錄」>「Organization Container」中。
4. 選取「JARVIS：導覽至「模型指令碼」、然後按一下「JARVIS快速入門」
5. 確認所有資產都能正常運作。
6. 尋找建立您自己應用程式的文件：PDF可在「模型指令碼」>「JARVIS文件」>「檔案瀏覽器」中找到。

針對零售使用案例自訂狀態和流程

您可以針對特定使用案例自訂對話管理程式的狀態和流程。在我們的零售範例中、我們有下列四個yaml檔案、可根據不同的意圖來引導對話。

請參閱下列每個檔案的檔案名稱和說明清單：

- 「MAIN_flow：yml」：定義主要對話流程和狀態、並在必要時將流程導向其他三個yaml檔案。
- 「REtail_flow。yml」：包含與零售或興趣點相關的狀態。系統會提供最近的零售店資訊、或是特定項目的價格。
- 「Weather（天氣）」流程：包含與天氣問題相關的狀態。如果無法判斷位置、系統會詢問後續問題以釐清。
- 「error_flow。yml」：處理使用者意圖不屬於上述三個yaml檔案的案例。顯示錯誤訊息後、系統會重新路由回接受使用者問題。下列各節包含這些yaml檔案的詳細定義。

main_flow

```
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
```

零售_流.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
```

We天氣_流.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
```

錯誤_流.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
```

將第三方API連線為履約引擎

我們將下列第三方API連線為履行引擎、以回答問題：

- "WeatherStack API"：返回給定位置的天氣、溫度、雨和雪。
- "Yelp Fusion 功能API"：傳回指定位置中最近的商店資訊。
- "eBay Python SDK"：傳回指定項目的價格。

NetApp零售助理示範

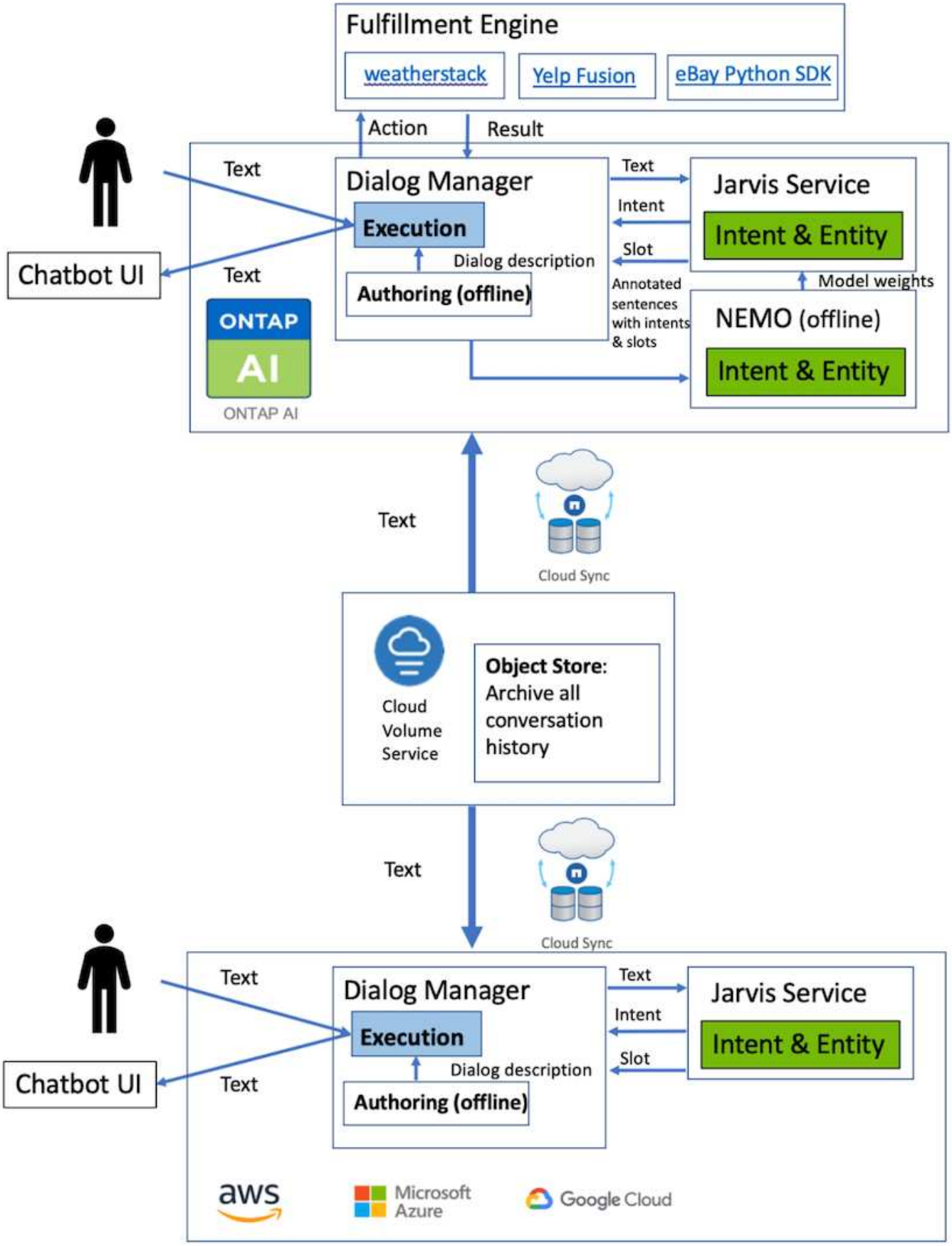
我們錄製了NetApp零售助理（Nara）的示範影片。

Nara 影片示範

Nara 影片示範

使用 NetApp BlueXP 複製與同步功能來歸檔對話記錄

我們可以每天將對話記錄傾印至 CSV 檔案一次、然後利用 BlueXP 複製與同步功能將記錄檔下載至本機儲存設備。下圖顯示 Jarvis 部署在內部部署和公有雲的架構、同時使用 BlueXP Copy and Sync 傳送 Nemo 訓練的對話記錄。如需Nemo訓練的詳細資料、請參閱一節 "使用Nemo訓練來擴充Intent模式"。


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訓練來擴充Intent模式

NVIDIA Nemo是NVIDIA專為建立對話式AI應用程式所打造的工具套件。此工具套件包含預先訓練的ASR、NLP和TS模組集合、可讓研究人員和資料科學家輕鬆建立複雜的神經網路架構、並將更多焦點放在設計自己的應用程式上。

如上例所示、Nara只能處理有限類型的問題。這是因為預先訓練的NLP模式只會訓練這些類型的問題。如果我們想要讓Nara能夠處理更廣泛的問題、就必須重新訓練自己的資料集。因此、我們在此示範如何使用Nemo來延伸NLP模式、以滿足需求。首先、我們會將從Nara收集的記錄轉換成Nemo格式、然後與資料集一起訓練、以強化NLP模式。

模型

我們的目標是讓Nara根據使用者偏好來排序項目。例如、我們可能會請Nara推薦評價最高的壽司餐廳、或是想要Nara以最低價格尋找這款牛仔褲。為此、我們使用Nemo提供的意向偵測和插槽填滿模式做為訓練模式。此模式可讓Nara瞭解搜尋偏好的意圖。

資料準備

為了訓練模型、我們會收集此類問題的資料集、並將其轉換為Nemo格式。在此列出我們用來訓練模型的檔案。

dict.intents.csv

此檔案列出我們希望Nemo瞭解的所有目標。在此、我們有兩個主要目的和一個意圖、只用於將不符合任何主要目的的問題分類。

```
price_check
find_the_store
unknown
```

dict.slots.csv

此檔案列出我們訓練問題上可標示的所有插槽。

```
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
```

訓練：**tsv**

這是主要的訓練資料集。每一行開頭都是檔案dict.intent.csv中列出的Intent類別之後的問題。從零開始列舉標籤。

訓練插槽**.tsv**

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

訓練模型

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

然後使用下列命令啟動容器。在此命令中、我們限制容器使用單一GPU（GPU ID = 1）、因為這是輕量化的訓練練習。我們也會將本機工作區/Works/nemo/對應至Container /nemo內的資料夾。

```
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
```

在容器內、如果我們想要從原本訓練好的Bert模型開始、可以使用下列命令來開始訓練程序。data_dir是設定訓練資料路徑的引數。Work目錄可讓您設定儲存檢查點檔案的位置。

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

如果我們有新的訓練資料集、而且想要改善先前的模式、我們可以使用下列命令、從停止點繼續進行。Checkpoint目錄會將路徑移至先前的檢查點資料夾。

```
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
```

推斷模型

我們需要在經過一定次數的時間後、驗證受過訓練的模型的效能。下列命令可讓我們逐一測試查詢。舉例來說、在這個命令中、我們想要檢查我們的模式是否能正確識別查詢「哪裡可以找到最好的義大利麵」的意圖。

```
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
```

接著、以下是推斷的輸出。在輸出中、我們可以看到我們訓練過的模型能夠正確預測future_the_store的意圖、並傳回我們感興趣的關鍵字。有了這些關鍵字、我們就能讓Nara搜尋使用者想要的內容、並進行更精確的搜尋。

```
[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
```