Jarvis, BlueXP Copy and Sync, Nemo를 사용하여 가상 어시스턴트를 구축합니다

NetApp Solutions

NetApp
April 20, 2024

목차

개요

이 섹션에서는 Virtual Retail Assistant 구현에 대해 자세히 설명합니다.

Jarvis 배포

에 등록할 수 있습니다 "Jarvis Early Access 프로그램" NGC(NVIDIA GPU Cloud)에서 Jarvis 컨테이너에 액세스 NVIDIA로부터 자격 증명을 받은 후 다음 단계를 사용하여 Jarvis를 배포할 수 있습니다.

1. NGC에 로그인합니다.
2. NGC를 통해 조직을 "ea-2-Jarvis"로 설정합니다.
3. Jarvis EA v0.2 자산 찾기: Jarvis 컨테이너는 '개인 레지스트리'>'조직 컨테이너'에 있습니다.
4. 자비스(Jarvis) 를 선택하고 모델 스크립트(Model Scripts) 로 이동한 다음 자비스 빠른 시작(Jarvis Quick Start) 을 클릭합니다
5. 모든 자산이 제대로 작동하는지 확인합니다.
6. PDF는 모델 스크립트 > Jarvis Documentation > File Browser에서 찾을 수 있습니다.

소매 사용 사례에 대한 상태 및 흐름을 사용자 지정합니다

특정 사용 사례에 맞게 Dialog Manager의 상태 및 흐름을 사용자 지정할 수 있습니다. 당사의 소매 예시에서 다음과 같은 네 가지 YAML 파일이 다양한 인도에 따라 대화를 유도합니다.

다음 파일 이름 목록과 각 파일에 대한 설명을 따르십시오.

- main_flow.yml: 주요 대화의 흐름과 상태를 정의하고 필요에 따라 다른 3개의 YAML 파일로 흐름을 안내합니다.
- RETail_flow.yml: 소매 또는 관심 지점 질문과 관련된 주가 포함되어 있습니다. 시스템은 가장 가까운 매장의 정보 또는 지정된 품목의 가격을 제공합니다.
- 날씨 흐름.yml: 날씨 문제와 관련된 주가 포함되어 있습니다. 위치를 확인할 수 없는 경우 시스템은 추가 질문을 통해 명확히 합니다.
- error_flow.yml: 위의 3가지 YAML 파일에 포함되지 않는 경우를 처리합니다. 오류 메시지를 표시한 후 시스템은 사용자 질문 수락으로 다시 라우팅합니다. 다음 섹션에는 이러한 YAML 파일에 대한 자세한 정의가 나와 있습니다.

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
```

WATEER_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
```

타사 API에 이행 엔진으로 연결합니다

다음 타사 API를 이행 엔진으로 연결하여 질문에 답변했습니다.

- "WeatherStack API를 참조하십시오": 지정된 위치에 날씨, 온도, 강우량 및 눈을 반환합니다.
- "Yelp Fusion API를 참조하십시오": 지정된 위치에서 가장 가까운 매장 정보를 반환합니다.
- "eBay Python SDK": 지정된 항목의 가격을 반환합니다.

NetApp Retail Assistant 데모

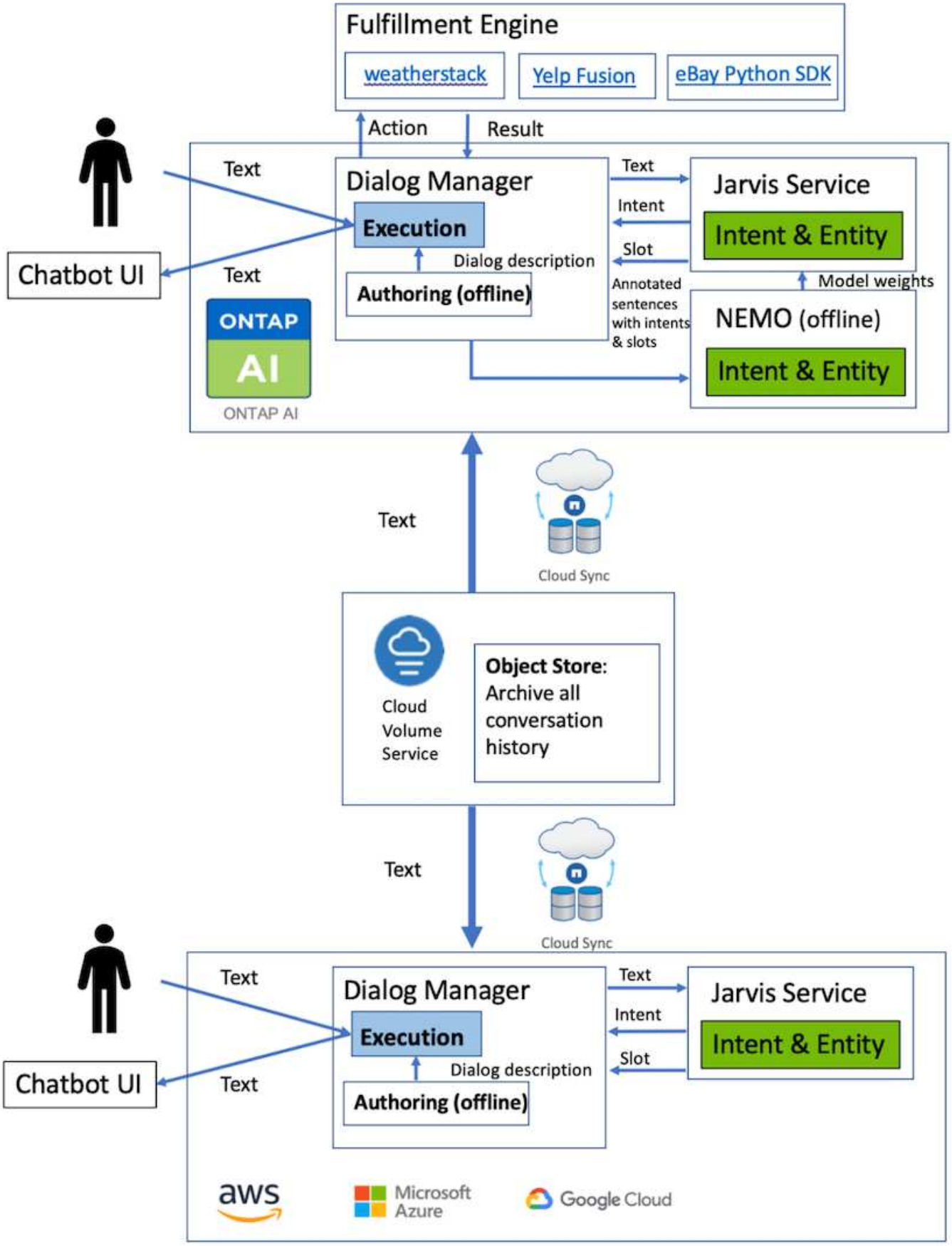
NetApp Retail Assistant(Nara)의 데모 비디오를 녹화했습니다.

나라 동영상 데모

나라 동영상 데모

NetApp BlueXP 복사 및 동기화를 사용하여 대화 기록을 아카이브합니다

하루에 한 번 대화 기록을 CSV 파일에 덤프하면 BlueXP 복사본 및 동기화 를 활용하여 로그 파일을 로컬 스토리지에 다운로드할 수 있습니다. 다음 그림에서는 Jarvis가 온프레미스 및 퍼블릭 클라우드에 구축하면서 BlueXP Copy 및 Sync를 사용하여 Nemo 교육을 위한 대화 기록을 전송하는 아키텍처를 보여 줍니다. Nemo 교육에 대한 자세한 내용은 섹션을 참조하십시오 "Nemo Training을 사용하여 Intent 모델을 확장합니다".


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

Nemo Training을 사용하여 Intent 모델을 확장합니다

NVIDIA Nemo는 NVIDIA에서 대화형 AI 애플리케이션을 만들기 위해 만든 툴킷입니다. 이 툴킷에는 ASR, NLP 및 TTS에 대한 사전 교육 모듈 모음이 포함되어 있어 연구자와 데이터 과학자가 복잡한 신경망 아키텍처를 쉽게 구성하고 자체 애플리케이션을 설계하는 데 더 많은 노력을 집중할 수 있습니다.

앞의 예에서와 같이 나라에서는 제한된 질문 유형만 처리할 수 있습니다. 사전 교육 받은 NLP 모델은 이러한 유형의 질문에만 교육을 제공하기 때문입니다. Nara가 보다 광범위한 질문을 처리하도록 하려면 자체 데이터세트를 사용하여 재교육해야 합니다. 따라서 여기서는 Nemo를 사용하여 NLP 모델을 확장하여 요구 사항을 충족하는 방법을 보여 줍니다. 우선 Nara에서 수집한 로그를 Nemo 형식으로 변환한 다음 NLP 모델을 향상시키기 위해 데이터 세트를 사용하여 훈련합니다.

모델

우리의 목표는 Nara가 사용자 선호도에 따라 항목을 정렬할 수 있도록 하는 것입니다. 예를 들어, 나라에게 최고 등급의 스시 레스토랑을 추천하거나 나라(Nara)가 가장 낮은 가격으로 청바지를 찾아보길 원할 수도 있습니다. 이를 위해 Nemo에 제공된 intent detection 및 slot filling 모델을 실습 모델로 사용한다. 이 모델을 통해 Nara는 선호하는 검색의 의도를 이해할 수 있습니다.

데이터 준비

모델을 학습하기 위해 이 유형의 질문에 대한 데이터 세트를 수집하고 이를 Nemo 형식으로 변환합니다. 여기서는 모델을 훈련하는 데 사용하는 파일을 나열했습니다.

dict.intents.csv

이 파일에는 Nemo가 이해할 수 있는 모든 인텐트가 나열되어 있습니다. 여기서는 일차 연고 2개와 일차 연고 중 하나에 적합하지 않은 질문을 분류하는 데만 사용되는 의도로 1개를 사용합니다.

```
price_check
find_the_store
unknown
```

dict.slots.csv

이 파일에는 교육 질문에 표시할 수 있는 모든 슬롯이 나열되어 있습니다.

```
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
```

훈련.**TSV**

이 데이터 세트는 주요 교육 데이터 세트입니다. 각 줄은 dict.intent.csv 파일의 의도 범주 목록에 따라 질문으로 시작합니다. 레이블은 0부터 열거됩니다.

기차_슬롯.**TSV**

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

모델 훈련

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

그런 다음 다음 다음 다음 다음 명령을 사용하여 컨테이너를 시작합니다. 이 명령은 간단한 교육 연습이므로 컨테이너가 단일 GPU(GPU ID=1)를 사용하도록 제한합니다. 또한 로컬 작업 공간/작업 공간/Nemo/를 컨테이너/Nemo 내부의 폴더에 매핑합니다.

```
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
```

컨테이너 내부에서 사전 훈련된 원래 BERT 모델에서 시작하려면 다음 명령을 사용하여 교육 절차를 시작할 수 있습니다. data_dir은 교육 데이터의 경로를 설정하기 위한 인수입니다. Work_dir 체크포인트 파일을 저장할 위치를 구성할 수 있습니다.

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

새로운 교육 데이터 세트가 있고 이전 모델을 개선하려는 경우 다음 명령을 사용하여 중지한 시점부터 계속 진행할 수 있습니다. checkpoint_dir 은 경로를 이전 체크포인트 폴더로 가져갑니다.

```
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
```

모델을 추론합니다

특정 수의 Epoch 후에 교육 이수 모델의 성능을 검증해야 합니다. 다음 명령을 사용하여 쿼리를 하나씩 테스트할 수 있습니다. 예를 들어, 이 명령에서 모델이 '최고의 파스타를 어디서 얻을 수 있는지'라는 질의의 의도를 제대로 파악할 수 있는지 확인해야 합니다.

```
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
```

그런 다음, 추론의 출력입니다. 출력물에서는 숙련된 모델이 find_the_store의 의도를 적절히 예측하고 관심 있는 키워드를 반환할 수 있습니다. 이러한 키워드를 사용하여 Nara는 사용자가 원하는 것을 검색하고 보다 정확한 검색을 수행할 수 있습니다.

```
[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
```