Erstellen Sie einen virtuellen Assistenten mit Jarvis, BlueXP Copy and Sync und Nemo

NetApp Solutions

NetApp
October 31, 2024

Inhalt

Erstellen Sie einen virtuellen Assistenten mit Jarvis, BlueXP Copy and Sync und Nemo

Überblick

In diesem Abschnitt wird die Implementierung des virtuellen Einzelhandels-Assistenten ausführlich beschrieben.

Jarvis-Bereitstellung

Sie können sich anmelden für "Jarvis Early Access-Programm" Um Zugriff auf Jarvis-Container auf NVIDIA GPU Cloud (NGC) zu erhalten. Nachdem Sie die Anmeldedaten von NVIDIA erhalten haben, können Sie Jarvis mithilfe der folgenden Schritte bereitstellen:

1. Anmelden bei NGC
2. Setzen Sie Ihr Unternehmen auf NGC: `ea-2-jarvis`.
3. Lokalisieren Sie Jarvis EA v0.2 Assets: Jarvis Container sind in `Private Registry` > `Organization Containers`.
4. Wählen Sie Jarvis: Navigieren Sie zu `Model Scripts` Und klicken `Jarvis Quick Start`
5. Überprüfen Sie, ob alle Ressourcen ordnungsgemäß funktionieren.
6. Finden Sie die Dokumentation zum Erstellen Ihrer eigenen Anwendungen: PDFs finden Sie unter `Model Scripts` > `Jarvis Documentation` > `File Browser`.

Passen Sie die Staaten und Abläufe für den Einzelhandel an

Sie können die Zustände und Abläufe des Dialog-Managers für Ihre spezifischen Anwendungsfälle anpassen. In unserem Einzelhandelbeispiel haben wir die folgenden vier yaml-Dateien, um das Gespräch nach verschiedenen Absichten zu lenken.

SE die folgende Liste mit Dateinamen und Beschreibung jeder Datei:

- `main_flow.yml`: Definiert die wichtigsten Gesprächsflüsse und -Zustände und leitet den Fluss zu den anderen drei yaml-Dateien, wenn nötig.
- `retail_flow.yml`: Enthält Staaten im Zusammenhang mit Einzelhandels- oder Interessenfragen. Das System liefert entweder die Informationen zum nächstgelegenen Geschäft oder den Preis eines bestimmten Artikels.
- `weather_flow.yml`: Enthält Staaten im Zusammenhang mit Wetterfragen. Wenn der Standort nicht ermittelt werden kann, fragt das System eine weitere Frage zur Klärung.
- `error_flow.yml`: Behandelt Fälle, in denen Benutzer Intents nicht in die oben genannten drei yaml-Dateien fallen. Nach dem Anzeigen einer Fehlermeldung wird das System erneut auf Benutzerfragen zurückleitet.die folgenden Abschnitte enthalten die detaillierten Definitionen für diese yaml-Dateien.

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
```

Wetter_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
```

Stellen Sie eine Verbindung zu APIs von Drittanbietern als Fulfillment Engine her

Wir haben die folgenden Drittanbieter-APIs als Fulfillment Engine verbunden, um Fragen zu beantworten:

- "WeatherStack API": Gibt das Wetter, die Temperatur, den Niederschlag und den Schnee an einem bestimmten Ort zurück.
- "Yelp Fusion-API": Gibt die nächstgelegenen Informationen an einem bestimmten Ort zurück.
- "EBay Python SDK": Gibt den Preis eines bestimmten Artikels zurück.

Demonstration des NetApp Retail Assistant

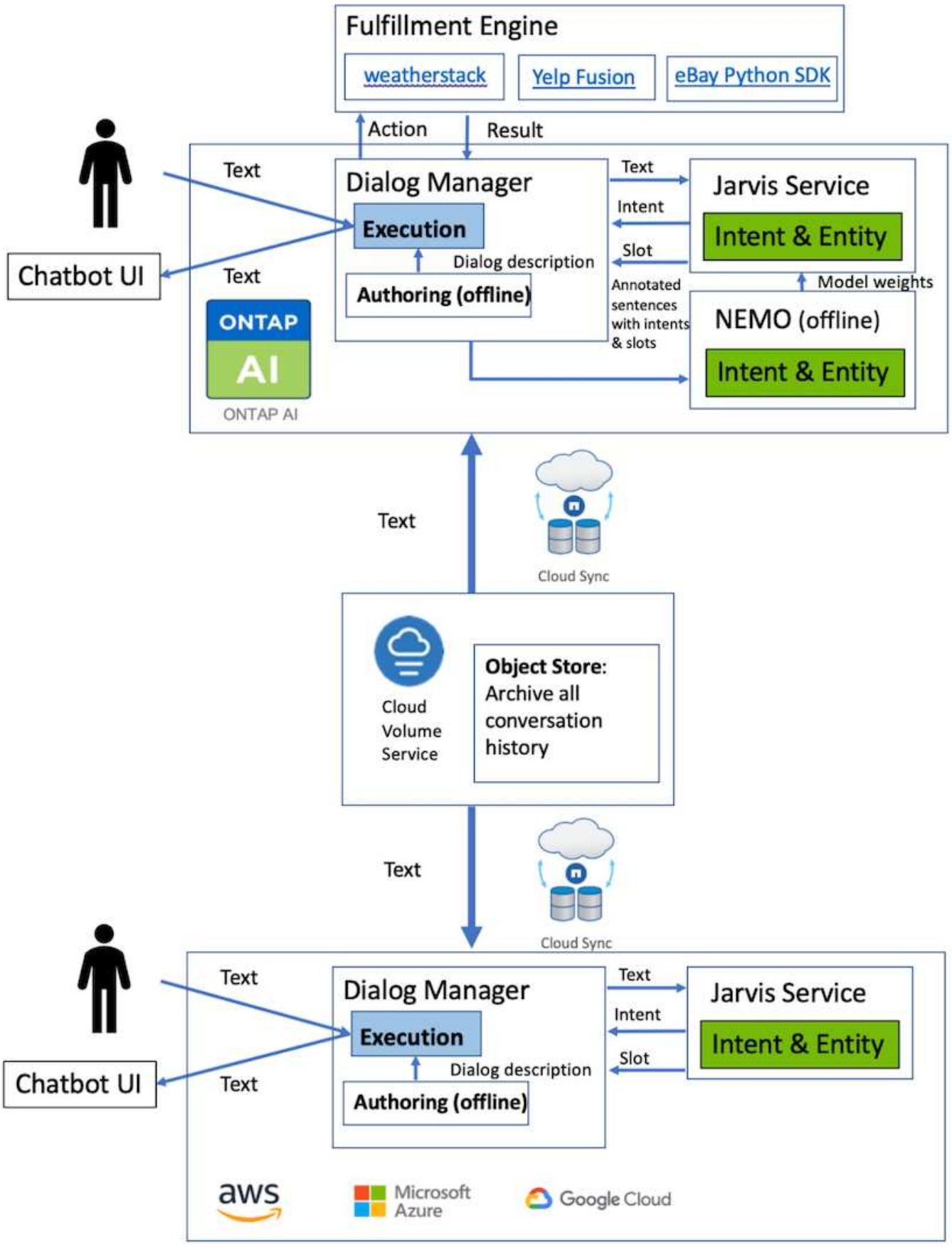
Wir haben ein Demovideo zu dem NetApp Retail Assistant (NARA) aufgenommen.

Video-Demonstration VON NARA

Video-Demonstration VON NARA

Verwenden Sie NetApp BlueXP, um den Gesprächsverlauf zu archivieren

Indem wir einmal täglich den Gesprächsverlauf in eine CSV-Datei kopieren, können wir die Protokolldateien mithilfe von BlueXP Copy and Sync im lokalen Storage herunterladen. Die folgende Abbildung zeigt die Architektur, in der Jarvis lokal und in Public Clouds implementiert wird und mit BlueXP Copy und Sync Konversationsverlauf beim Nemo Training übertragen wird. Einzelheiten zum Nemo Training finden Sie im Abschnitt "Erweitern Sie die Intent-Modelle mit Nemo Training".


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

Erweitern Sie die Intent-Modelle mit Nemo Training

NVIDIA Nemo ist ein von NVIDIA entwickeltes Toolkit für die Erstellung von umgangssprachlichen KI-Applikationen. Dieses Toolkit enthält vortrainierte Module für ASR, NLP und TTS, mit denen Forscher und Datenwissenschaftler komplexe neuronale Netzarchitekturen leicht erstellen und sich stärker auf die Entwicklung ihrer eigenen Anwendungen konzentrieren können.

Wie im vorherigen Beispiel gezeigt, kann NARA nur eine begrenzte Art von Fragen verarbeiten. Denn das vortrainierte NLP-Modell trainiert nur bei diesen Fragen. Wenn NARA in der Lage ist, ein breiteres Spektrum von Fragen zu beantworten, müssen wir diese mit unseren eigenen Datensätzen neu Schulen. So zeigen wir hier, wie wir mit Nemo das NLP-Modell erweitern können, um den Anforderungen gerecht zu werden. Wir beginnen damit, das aus NARA gesammelte Protokoll in das Format für Nemo umzuwandeln und dann mit dem Datensatz zu trainieren, um das NLP-Modell zu verbessern.

Modell

Unser Ziel ist ES, NARA in die Lage zu versetzen, die Elemente anhand der Benutzereinstellungen zu sortieren. Zum Beispiel könnten wir NARA bitten, das beste Sushi-Restaurant zu empfehlen oder MÖCHTEN, DASS NARA die Jeans mit dem niedrigsten Preis nachschlagen kann. Dazu verwenden wir das in Nemo bereitgestellte Absichtserkennungs- und Steckplatzfüllmodell als Trainingsmodell. Mit diesem Modell KANN NARA den Zweck der Suchpräferenz verstehen.

Datenvorbereitung

Um das Modell zu trainieren, sammeln wir den Datensatz für diese Art von Frage und konvertieren ihn in das Nemo-Format. Hier haben wir die Dateien aufgelistet, die wir für das Training des Modells verwenden.

dict.intents.csv

Diese Datei enthält alle Absichten, die wir von Nemo verstehen möchten. Hier haben wir zwei primäre Absichten und eine Absicht, die nur zur Kategorisierung der Fragen verwendet werden, die nicht in eine der primären Absichten passen.

```
price_check
find_the_store
unknown
```

dict.slots.csv

Diese Datei enthält alle Slots, die wir bei unseren Trainingsfragen kennzeichnen können.

```
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
```

Zug.tsv

Dies ist der Haupt-Trainingsdatensatz. Jede Zeile beginnt mit der Frage nach der Liste der Absichtskategorie in der Datei dict.intent.csv. Die Bezeichnung wird ab Null aufgezählt.

Train_slots.tsv

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

Trainieren Sie das Modell

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

Wir verwenden dann den folgenden Befehl, um den Container zu starten. In diesem Befehl begrenzen wir den Container auf eine einzelne GPU (GPU-ID = 1), da es sich hierbei um eine leichte Trainingsübung handelt. Wir ordnen unseren lokalen Arbeitsbereich /Workspace/nemo/ auch dem Ordner im Container /nemo zu.

```
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
```

Im Container können wir, wenn wir von dem ursprünglich vortrainierten BERT-Modell starten möchten, den folgenden Befehl verwenden, um das Training zu starten. Data_dir ist das Argument, um den Pfad der Trainingsdaten einzurichten. Mit Work_dir können Sie festlegen, wo Sie die Checkpoint-Dateien speichern möchten.

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

Wenn wir neue Trainingsdatensätze haben und das vorherige Modell verbessern möchten, können wir mit dem folgenden Befehl fortfahren, nachdem wir das Ende gestoppt haben. Checkpoint_dir führt den Pfad zum Ordner frühere Checkpoints.

```
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
```

Inferenz: Das Modell

Wir müssen die Performance des trainierten Modells nach einer bestimmten Anzahl von Epoch-Durchläufen validieren. Mit dem folgenden Befehl können wir die Abfrage One-by-One testen. In diesem Befehl möchten wir beispielsweise überprüfen, ob unser Modell die Absicht der Abfrage richtig identifizieren kann `where can I get the best pasta`.

```
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
```

Dann ist die Ausgabe des Inferenz unten. In der Ausgabe sehen wir, dass unser trainiertes Modell die Absicht Find_the_Store richtig vorhersagen kann und die Schlüsselworte, die wir interessieren, zurückgeben kann. Mit diesen Stichwörtern ermöglichen wir dem NARA, nach dem zu suchen, was der Benutzer wünscht, und führen eine präzisere Suche durch.

```
[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
```