Crea un asistente virtual con Jarvis, BlueXP Copy and Sync y Nemo

NetApp Solutions

NetApp
April 26, 2024

Tabla de contenidos

Descripción general

En esta sección se ofrece información detallada sobre la implantación del asistente de venta al por menor virtual.

Despliegue de Jarvis

Puede registrarse para "Programa Jarvis Early Access" Para obtener acceso a contenedores Jarvis en NVIDIA GPU Cloud (NGC). Después de recibir credenciales de NVIDIA, puede implementar Jarvis siguiendo los pasos siguientes:

1. Firma a NGC.
2. Establezca la organización en NGC: `ea-2-jarvis`.
3. Localice Jarvis EA v0.2 activos: Contenedores Jarvis están en `Private Registry` > `Organization Containers`.
4. Seleccione Jarvis: Desplácese a. `Model Scripts` y haga clic en `Jarvis Quick Start`
5. Compruebe que todos los activos funcionan correctamente.
6. Busque la documentación para crear sus propias aplicaciones: Los archivos PDF se pueden encontrar en `Model Scripts` > `Jarvis Documentation` > `File Browser`.

Personalizar Estados y flujos para el caso de uso de comercio minorista

Puede personalizar los Estados y flujos de Dialog Manager para sus casos de uso específicos. En nuestro ejemplo de venta al por menor, tenemos los siguientes cuatro archivos yaml para dirigir la conversación según diferentes intentos.

S la siguiente lista de nombres de archivo y descripción de cada archivo:

- `main_flow.yml`: Define los principales flujos y estados de conversación y dirige el flujo a los otros tres archivos yaml cuando sea necesario.
- `retail_flow.yml`: Contiene estados relacionados con preguntas sobre puntos de interés o minoristas. El sistema proporciona la información de la tienda más cercana o el precio de un artículo determinado.
- `weather_flow.yml`: Contiene estados relacionados con las preguntas sobre el clima. Si no se puede determinar la ubicación, el sistema hace una pregunta de seguimiento para aclarar.
- `error_flow.yml`: Trata los casos en los que las intenciones del usuario no entran en los tres archivos yaml anteriores. Después de mostrar un mensaje de error, el sistema vuelve a enruta para aceptar preguntas de usuario.las siguientes secciones contienen las definiciones detalladas de estos archivos yaml.

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
```

Conéctese a API de terceros como motor de cumplimiento

Conectamos las siguientes API de terceros como motor de cumplimiento de normativas para responder a las preguntas:

- "API de WeatherStack": regresa el clima, la temperatura, las precipitaciones y la nieve en un lugar determinado.
- "API de Fusion yelp": devuelve la información de la tienda más cercana en una ubicación determinada.
- "SDK de Python de eBay": devuelve el precio de un artículo determinado.

Demostración del asistente de venta al por menor de NetApp

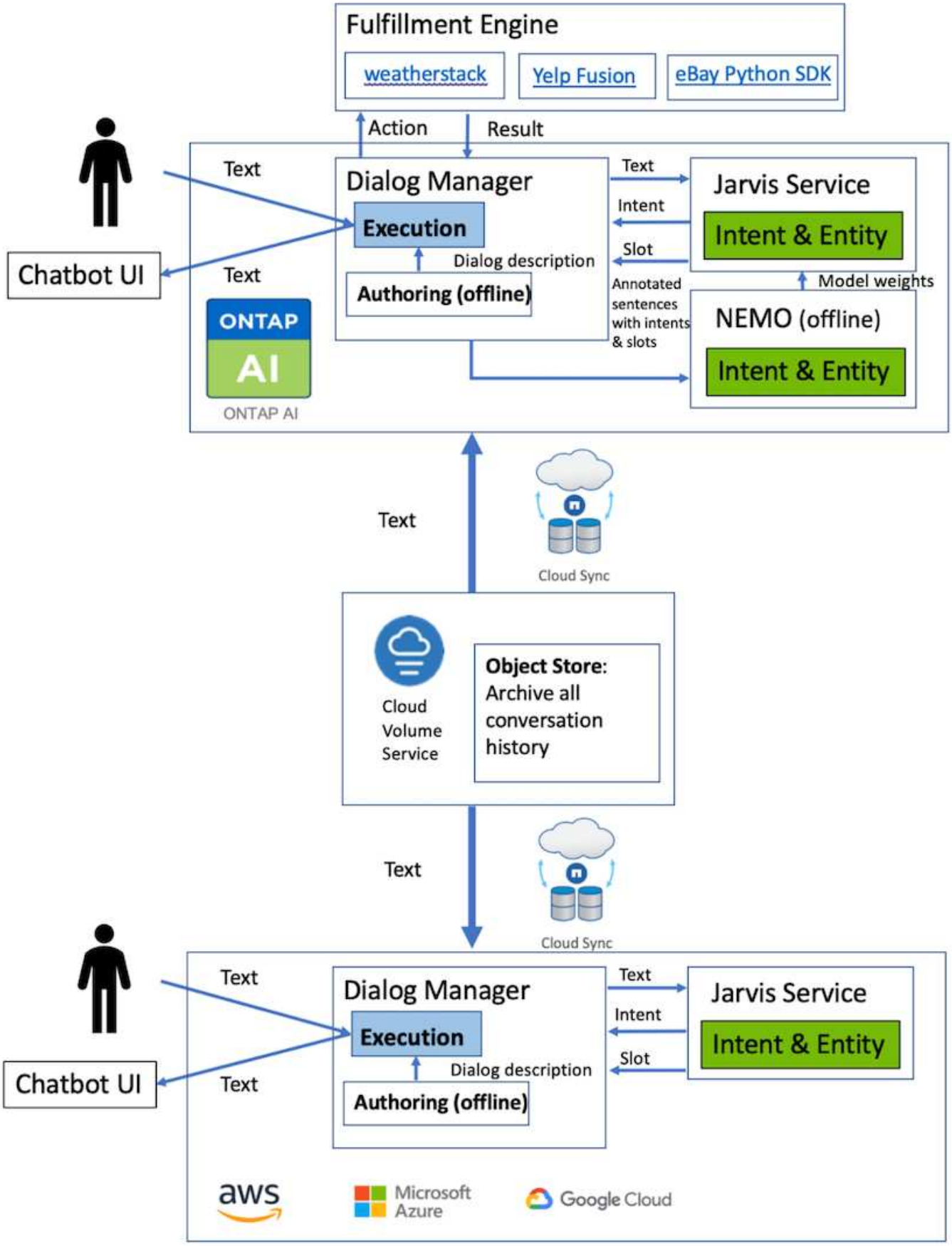
Grabamos un vídeo de demostración del Asistente para minoristas de NetApp (NARA).

Demostración en vídeo de NARA

Demostración en vídeo de NARA

Utiliza la copia y sincronización de NetApp BlueXP para archivar el historial de conversaciones

Al volcar el historial de conversaciones en un archivo CSV una vez al día, podemos aprovechar la copia y sincronización de BlueXP para descargar los archivos de registro en un almacenamiento local. La siguiente figura muestra la arquitectura que indica que Jarvis ha puesto en marcha on-premises y en nubes públicas, mientras utiliza la función Copy y Sync de BlueXP para enviar el historial de conversaciones para la formación de Nemo. En la sección encontrará más información sobre la formación de Nemo "Expanda modelos de intención utilizando Nemo Training".


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

Expanda modelos de intención utilizando Nemo Training

NVIDIA Nemo es un kit de herramientas creado por NVIDIA para crear aplicaciones de IA conversacionales. Este kit de herramientas incluye colecciones de módulos preentrenados para ASR, NLP y TTS, lo que permite a investigadores y científicos de datos componer fácilmente arquitecturas complejas de redes neuronales y centrarse más en el diseño de sus propias aplicaciones.

Como se muestra en el ejemplo anterior, NARA sólo puede manejar un tipo limitado de preguntas. Esto se debe a que el modelo NLP pre-entrenado sólo entrena en este tipo de preguntas. Si queremos permitir QUE NARA pueda gestionar una gama más amplia de preguntas, debemos volver a formar este con nuestros propios conjuntos de datos. Por lo tanto, aquí mostramos cómo podemos utilizar Nemo para ampliar el modelo NLP para satisfacer los requisitos. Comenzamos convirtiendo el registro recolectado de NARA en el formato de Nemo, y luego entrenamos con el conjunto de datos para mejorar el modelo NLP.

Modelo

Nuestro objetivo es permitir A NARA ordenar los elementos según las preferencias del usuario. Por ejemplo, podemos pedir A NARA que sugiera el restaurante de sushi con mejor calificación o que quiera QUE NARA busque los vaqueros con el precio más bajo. Para ello, utilizamos el modelo de detección de intención y relleno de ranuras proporcionado en Nemo como modelo de entrenamiento. Este modelo permite A NARA comprender la intención de buscar preferencias.

Preparación de datos

Para entrenar el modelo, recopilamos el conjunto de datos para este tipo de preguntas y lo convertimos al formato Nemo. Aquí enumeramos los archivos que utilizamos para entrenar el modelo.

dict.intents.csv

Este archivo enumera todos los intentos que queremos que el Nemo entienda. Aquí tenemos dos intentos principales y una intención que sólo se utiliza para categorizar las preguntas que no encajan en ninguno de los intentos primarios.

```
price_check
find_the_store
unknown
```

dict.slots.csv

En este archivo se enumeran todas las ranuras que podemos etiquetar en nuestras preguntas de formación.

```
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
```

train.tsv

Este es el conjunto de datos de entrenamiento principal. Cada línea comienza con la pregunta que sigue a la lista de la categoría de intención en el archivo dict.intent.csv. La etiqueta se enumera a partir de cero.

train_slots.tsv

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

Entrenar el modelo

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

A continuación, utilizamos el siguiente comando para iniciar el contenedor. En este comando, limitamos el contenedor para usar una única GPU (ID de GPU = 1), ya que se trata de un ejercicio de entrenamiento de poco peso. También mapeamos nuestro espacio de trabajo local /Workspace/nemo/ a la carpeta dentro del contenedor /nemo.

```
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
```

Dentro del contenedor, si queremos empezar desde el modelo ORIGINAL BERT pre-entrenado, podemos usar el siguiente comando para iniciar el procedimiento de entrenamiento. data_dir es el argumento para establecer la ruta de los datos de entrenamiento. dir_trabajo le permite configurar dónde desea almacenar los archivos de punto de control.

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

Si contamos con nuevos conjuntos de datos de entrenamiento y queremos mejorar el modelo anterior, podemos utilizar el siguiente comando para continuar desde el punto que hemos detenido. checkpoint_dir lleva la ruta a la carpeta de puntos de control anteriores.

```
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
```

Inferencia del modelo

Se debe validar el rendimiento del modelo entrenado después de una serie determinada de épocas. El siguiente comando nos permite probar la consulta una por una. Por ejemplo, en este comando, queremos comprobar si nuestro modelo puede identificar adecuadamente la intención de la consulta `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
```

A continuación, se muestra la salida de la inferencia. En el resultado, podemos ver que nuestro modelo entrenado puede predecir correctamente la intención find_the_store, y devolver las palabras clave en las que estamos interesados. Con estas palabras clave, permitimos A LA NARA buscar lo que los usuarios desean y realizar una búsqueda más precisa.

```
[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
```