



Créez un assistant virtuel à l'aide de Jarvis, de BlueXP Copy and Sync et de Nemo

NetApp Solutions

NetApp
April 26, 2024

This PDF was generated from https://docs.netapp.com/fr-fr/netapp-solutions/ai/cainvidia_jarvis_deployment.html on April 26, 2024. Always check docs.netapp.com for the latest.

Sommaire

- Présentation 1
 - Jarvis déploiement 1
 - Personnalisez les États et les flux pour le cas d'utilisation Vente au détail 1
 - Connectez-vous à des API tierces en tant que moteur de traitement 9
 - Démonstration de NetApp Retail Assistant 9
 - Utilisez la copie et la synchronisation NetApp BlueXP pour archiver l'historique des conversations 10
 - Développez les modèles d'intention à l'aide de la formation Nemo 12

Présentation

Cette section fournit des détails sur la mise en œuvre de l'assistant de vente au détail virtuel.

Jarvis déploiement

Vous pouvez vous inscrire à "[Jarvis accès précoce](#)" Pour accéder aux conteneurs Jarvis sur NVIDIA GPU Cloud (NGC). Après avoir reçu les informations d'identification de NVIDIA, vous pouvez déployer Jarvis en procédant comme suit :

1. Connexion au contrôleur NGC.
2. Définissez votre organisation sur NGC : `ea-2-jarvis`.
3. Localiser Jarvis EA v0.2 biens: Les conteneurs Jarvis sont dans `Private Registry > Organization Containers`.
4. Sélectionnez Jarvis : accédez à `Model Scripts` et cliquez sur `Jarvis Quick Start`
5. Vérifiez que toutes les ressources fonctionnent correctement.
6. Trouvez la documentation pour créer vos propres applications : les PDF se trouvent dans `Model Scripts > Jarvis Documentation > File Browser`.

Personnalisez les États et les flux pour le cas d'utilisation Vente au détail

Vous pouvez personnaliser les États et les flux de Dialog Manager pour vos cas d'utilisation spécifiques. Dans notre exemple de vente au détail, nous avons les quatre fichiers yaml suivants pour diriger la conversation selon différentes intentions.

Se la liste suivante des noms de fichier et la description de chaque fichier :

- `main_flow.yml`: Définit les principaux flux et États de conversation et dirige le flux vers les trois autres fichiers yaml si nécessaire.
- `retail_flow.yml`: Contient des États liés à des questions de détail ou de points d'intérêt. Le système fournit soit les informations du magasin le plus proche, soit le prix d'un article donné.
- `weather_flow.yml`: Contient des États relatifs aux questions météorologiques. Si l'emplacement ne peut pas être déterminé, le système pose une question de suivi pour clarifier.
- `error_flow.yml`: Gère les cas où les intentions des utilisateurs ne tombent pas dans les trois fichiers yaml ci-dessus. Après l'affichage d'un message d'erreur, le système revient à accepter les questions de l'utilisateur. Les sections suivantes contiennent les définitions détaillées de ces fichiers 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
```

météo_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
```

Connectez-vous à des API tierces en tant que moteur de traitement

Nous avons connecté les API tierces suivantes en tant que moteur de traitement pour répondre aux questions :

- ["API WeatherStack"](#): renvoie la météo, la température, la pluie et la neige dans un endroit donné.
- ["API Yelp Fusion"](#): renvoie les informations de magasin les plus proches dans un emplacement donné.
- ["Kit de développement logiciel eBay Python"](#): renvoie le prix d'un article donné.

Démonstration de NetApp Retail Assistant

Nous avons enregistré une vidéo de démonstration de l'Assistant vente au détail NetApp (NARA).

Vidéo de démonstration de NARA

[Vidéo de démonstration de NARA](#)

NetApp NARA



Hi, welcome to NARA retail and weather service. How can I help you?

Write your message...

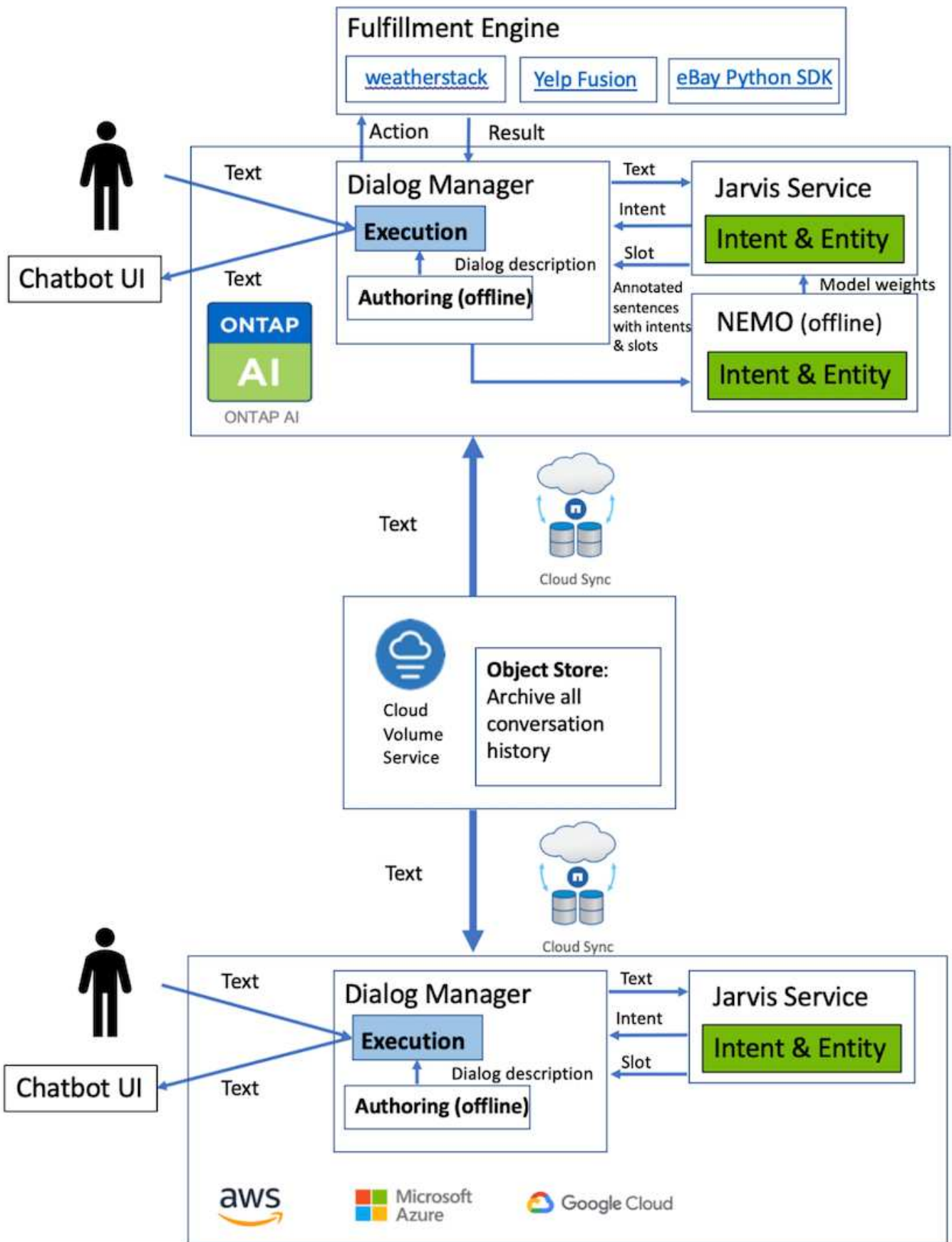
Submit

System replied. Waiting for user input.

Unmute System Speech

Utilisez la copie et la synchronisation NetApp BlueXP pour archiver l'historique des conversations

En vidant une fois par jour l'historique des conversations dans un fichier CSV, nous pouvons ensuite utiliser BlueXP Copy and Sync pour télécharger les fichiers journaux dans un système de stockage local. La figure suivante montre l'architecture du déploiement de Jarvis sur site et dans les clouds publics, tout en utilisant BlueXP Copy and Sync pour envoyer l'historique des conversations pour la formation Nemo. Les détails de la formation de Nemo sont disponibles dans la section ["Développez les modèles d'intention à l'aide de la formation Nemo"](#).



Développez les modèles d'intention à l'aide de la formation Nemo

NVIDIA Nemo est un kit conçu par NVIDIA pour créer des applications d'IA conversationnelles. Ce kit comprend des ensembles de modules pré-entraînés pour ASR, NLP et TTS, ce qui permet aux chercheurs et aux scientifiques des données de composer facilement des architectures de réseaux neuronaux complexes et de se concentrer davantage sur la conception de leurs propres applications.

Comme le montre l'exemple précédent, NARA ne peut traiter qu'un type limité de question. En effet, le modèle NLP pré-formé ne s'entraîne que sur ces types de questions. Si nous voulons permettre À NARA de gérer un plus large éventail de questions, nous devons le réentraîner avec nos propres jeux de données. Ainsi, ici, nous démontrons comment nous pouvons utiliser Nemo pour étendre le modèle NLP pour satisfaire les exigences. Nous commençons par convertir le journal collecté à partir DE NARA dans le format pour Nemo, puis nous entraînons avec le dataset pour améliorer le modèle NLP.

Modèle

Notre objectif est de permettre À NARA de trier les éléments en fonction des préférences de l'utilisateur. Par exemple, nous pourrions demander À NARA de proposer le restaurant de sushis le mieux noté ou pourrait vouloir NARA chercher les jeans avec le prix le plus bas. À cette fin, nous utilisons le modèle de détection d'intention et de remplissage de fente fourni dans Nemo comme modèle d'entraînement. Ce modèle permet À NARA de comprendre l'intention de la préférence de recherche.

Préparation des données

Pour entraîner le modèle, nous collectons l'ensemble de données pour ce type de question et le convertissons au format Nemo. Ici, nous avons répertorié les fichiers que nous utilisons pour entraîner le modèle.

dict.intents.csv

Ce fichier liste tous les éléments que nous voulons que le Nemo comprenne. Ici, nous avons deux intentions principales et une intention seulement utilisée pour classer les questions qui ne correspondent à aucune des intentions principales.

```
price_check
find_the_store
unknown
```

dict.slots.csv

Ce fichier répertorie tous les emplacements que nous pouvons étiqueter sur nos questions de formation.

```
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

Il s'agit du dataset d'entraînement principal. Chaque ligne commence par la question qui suit la liste des catégories d'intention dans le fichier dict.intent.csv. L'étiquette est énumérée à partir de zéro.

train_slots.tsv

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

Entraîner le modèle

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

Nous utilisons ensuite la commande suivante pour lancer le conteneur. Dans cette commande, nous limitons le conteneur à utiliser un seul GPU (ID de processeur graphique = 1), car il s'agit d'un exercice d'entraînement léger. Nous mappons également notre espace de travail local /Workspace/nemo/ vers le dossier à l'intérieur du conteneur /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
```

Dans le conteneur, si nous voulons commencer par le modèle original de BERT pré-formé, nous pouvons utiliser la commande suivante pour démarrer la procédure de formation. `data_dir` est l'argument pour définir le chemin des données d'entraînement. `work_dir` vous permet de configurer l'emplacement où vous souhaitez stocker les fichiers de point de contrôle.

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

Si nous avons de nouveaux datasets d'entraînement et que nous souhaitons améliorer le modèle précédent, nous pouvons utiliser la commande suivante pour continuer à partir du point que nous avons arrêté. `checkpoint_dir` indique le chemin d'accès au dossier points de contrôle précédent.


```
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
```

Inférence du modèle

Nous devons valider la performance du modèle entraîné après un certain nombre de tests. La commande suivante nous permet de tester la requête un par un. Par exemple, dans cette commande, nous voulons vérifier si notre modèle peut correctement identifier l'intention de la requête `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
```

Ensuite, le résultat suivant est le résultat de l'inférence. Dans ce résultat, nous pouvons constater que notre modèle entraîné peut prévoir correctement l'intention `Find_the_store` et renvoyer les mots-clés qui nous intéressent. Avec ces mots-clés, nous permettons à NARA de rechercher ce que les utilisateurs veulent et de faire une recherche plus précise.

```
[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
```

Informations sur le copyright

Copyright © 2024 NetApp, Inc. Tous droits réservés. Imprimé aux États-Unis. Aucune partie de ce document protégé par copyright ne peut être reproduite sous quelque forme que ce soit ou selon quelque méthode que ce soit (graphique, électronique ou mécanique, notamment par photocopie, enregistrement ou stockage dans un système de récupération électronique) sans l'autorisation écrite préalable du détenteur du droit de copyright.

Les logiciels dérivés des éléments NetApp protégés par copyright sont soumis à la licence et à l'avis de non-responsabilité suivants :

CE LOGICIEL EST FOURNI PAR NETAPP « EN L'ÉTAT » ET SANS GARANTIES EXPRESSES OU TACITES, Y COMPRIS LES GARANTIES TACITES DE QUALITÉ MARCHANDE ET D'ADÉQUATION À UN USAGE PARTICULIER, QUI SONT EXCLUES PAR LES PRÉSENTES. EN AUCUN CAS NETAPP NE SERA TENU POUR RESPONSABLE DE DOMMAGES DIRECTS, INDIRECTS, ACCESSOIRES, PARTICULIERS OU EXEMPLAIRES (Y COMPRIS L'ACHAT DE BIENS ET DE SERVICES DE SUBSTITUTION, LA PERTE DE JOUISSANCE, DE DONNÉES OU DE PROFITS, OU L'INTERRUPTION D'ACTIVITÉ), QUELLES QU'EN SOIENT LA CAUSE ET LA DOCTRINE DE RESPONSABILITÉ, QU'IL S'AGISSE DE RESPONSABILITÉ CONTRACTUELLE, STRICTE OU DÉLICTELLE (Y COMPRIS LA NÉGLIGENCE OU AUTRE) DÉCOULANT DE L'UTILISATION DE CE LOGICIEL, MÊME SI LA SOCIÉTÉ A ÉTÉ INFORMÉE DE LA POSSIBILITÉ DE TELS DOMMAGES.

NetApp se réserve le droit de modifier les produits décrits dans le présent document à tout moment et sans préavis. NetApp décline toute responsabilité découlant de l'utilisation des produits décrits dans le présent document, sauf accord explicite écrit de NetApp. L'utilisation ou l'achat de ce produit ne concède pas de licence dans le cadre de droits de brevet, de droits de marque commerciale ou de tout autre droit de propriété intellectuelle de NetApp.

Le produit décrit dans ce manuel peut être protégé par un ou plusieurs brevets américains, étrangers ou par une demande en attente.

LÉGENDE DE RESTRICTION DES DROITS : L'utilisation, la duplication ou la divulgation par le gouvernement sont sujettes aux restrictions énoncées dans le sous-paragraphe (b)(3) de la clause Rights in Technical Data-Noncommercial Items du DFARS 252.227-7013 (février 2014) et du FAR 52.227-19 (décembre 2007).

Les données contenues dans les présentes se rapportent à un produit et/ou service commercial (tel que défini par la clause FAR 2.101). Il s'agit de données propriétaires de NetApp, Inc. Toutes les données techniques et tous les logiciels fournis par NetApp en vertu du présent Accord sont à caractère commercial et ont été exclusivement développés à l'aide de fonds privés. Le gouvernement des États-Unis dispose d'une licence limitée irrévocable, non exclusive, non cessible, non transférable et mondiale. Cette licence lui permet d'utiliser uniquement les données relatives au contrat du gouvernement des États-Unis d'après lequel les données lui ont été fournies ou celles qui sont nécessaires à son exécution. Sauf dispositions contraires énoncées dans les présentes, l'utilisation, la divulgation, la reproduction, la modification, l'exécution, l'affichage des données sont interdits sans avoir obtenu le consentement écrit préalable de NetApp, Inc. Les droits de licences du Département de la Défense du gouvernement des États-Unis se limitent aux droits identifiés par la clause 252.227-7015(b) du DFARS (février 2014).

Informations sur les marques commerciales

NETAPP, le logo NETAPP et les marques citées sur le site <http://www.netapp.com/TM> sont des marques déposées ou des marques commerciales de NetApp, Inc. Les autres noms de marques et de produits sont des marques commerciales de leurs propriétaires respectifs.