Ai conversazionale con NVIDIA

NetApp Solutions

NetApp
April 26, 2024

Sommario

WP-7328: Ai di NetApp Conversational con NVIDIA Jarvis . . . 1
Panoramica della soluzione . . . 1
Tecnologia della soluzione . . . 3
Panoramica . . . 5
Conclusione . . . 19
Ringraziamenti . . . 20
Dove trovare ulteriori informazioni . . . 20

WP-7328: Ai di NetApp Conversational con NVIDIA Jarvis

Rick Huang, Sung-Han Lin, NetApp Davide Onofrio, NVIDIA

La famiglia di sistemi NVIDIA DGX è costituita dai primi sistemi al mondo basati su intelligenza artificiale integrata (ai) costruiti appositamente per l'ai aziendale. I sistemi storage NetApp AFF offrono performance estreme e funzionalità di gestione dei dati del cloud ibrido leader di settore. NetApp e NVIDIA hanno collaborato per creare l'architettura di riferimento ai di NetApp ONTAP, una soluzione chiavi in mano per i carichi di lavoro di ai e machine learning (ML) che offre performance, affidabilità e supporto di livello Enterprise.

Questo white paper fornisce una guida direzionale ai clienti che sviluppano sistemi di ai conversazionali a supporto di diversi casi di utilizzo in diversi mercati verticali del settore. Include informazioni sull'implementazione del sistema utilizzando NVIDIA Jarvis. I test sono stati eseguiti utilizzando una stazione NVIDIA DGX e un sistema storage NetApp AFF A220.

Il pubblico di riferimento per la soluzione comprende i seguenti gruppi:

- Enterprise Architect che progettano soluzioni per lo sviluppo di modelli di ai e software per casi di utilizzo conversa dell'ai, come un assistente virtuale al dettaglio
- Data scientist alla ricerca di modi efficienti per raggiungere gli obiettivi di sviluppo della modellazione linguistica
- Data engineer incaricati di gestire ed elaborare dati di testo come domande dei clienti e trascrizioni di dialoghi
- Dirigenti e responsabili delle decisioni IT e business leader interessati a trasformare l'esperienza di intelligenza artificiale conversazionale e a ottenere il più rapido time-to-market dalle iniziative di intelligenza artificiale

Panoramica della soluzione

NetApp ONTAP ai e BlueXP Copy and Sync

L'architettura NetApp ONTAP ai, basata su sistemi NVIDIA DGX e sistemi storage connessi al cloud, è stata sviluppata e verificata da NetApp e NVIDIA. Questa architettura di riferimento offre alle organizzazioni IT i seguenti vantaggi:

- Elimina le complessità di progettazione
- Consente una scalabilità indipendente di calcolo e storage
- Consente ai clienti di partire da piccoli e scalare perfettamente
- Offre una vasta gamma di opzioni di storage per diverse esigenze di performance e costi NetApp ONTAP ai integra perfettamente i sistemi DGX e i sistemi storage NetApp AFF A220 con networking all'avanguardia. I sistemi NetApp ONTAP ai e DGX semplificano le implementazioni ai eliminando la complessità e le congetture di progettazione. I clienti possono iniziare a crescere in maniera ininterrotta e allo stesso tempo gestire in modo intelligente i dati dall'edge al core, fino al cloud e viceversa.

NetApp BlueXP Copy e Sync ti permette di spostare facilmente i dati tra vari protocolli, tra due NFS share, due

CIFS share, oppure un file share e lo storage Amazon S3, Amazon Elastic file System (EFS) o Azure Blob. Il funzionamento Active-Active consente di continuare a lavorare contemporaneamente con l'origine e la destinazione, sincronizzando in modo incrementale le modifiche dei dati quando necessario. Consentendoti di spostare e sincronizzare in modo incrementale i dati tra qualsiasi sistema di origine e destinazione, sia on-premise che basato sul cloud, BlueXP Copy and Sync apre una vasta gamma di nuovi modi in cui puoi utilizzare i dati. La migrazione dei dati tra sistemi on-premise, cloud on-boarding e migrazione del cloud o collaboration e analytics dei dati diventa facilmente realizzabile. La figura seguente mostra le fonti e le destinazioni disponibili.

Nei sistemi ai conversazionali, gli sviluppatori possono sfruttare BlueXP Copy e Sync per archiviare la cronologia delle conversazioni dal cloud ai data center, consentendo il training offline dei modelli di elaborazione del linguaggio naturale (NLP). Attraverso modelli di training per riconoscere più intenti, il sistema di ai convergenti sarà meglio attrezzato per gestire domande più complesse da parte degli utenti finali.

Framework multimodale NVIDIA Jarvis

"NVIDIA Jarvis" È un framework end-to-end per la creazione di servizi di ai conversivi. Include i seguenti servizi ottimizzati per GPU:

- Riconoscimento vocale automatico (ASR)
- Comprensione del linguaggio naturale (NLU)
- Integrazione con servizi di adempimento specifici del dominio
- Text-to-speech (TTS)
- I servizi basati su computer Vision (CV) Jarvis utilizzano modelli di deep learning all'avanguardia per affrontare il complesso e impegnativo compito dell'ai conversazionale in tempo reale. Per consentire un'interazione naturale e in tempo reale con un utente finale, i modelli devono completare il calcolo in meno di 300 millisecondi. Le interazioni naturali sono impegnative e richiedono un'integrazione sensoriale multimodale. Anche le pipeline dei modelli sono complesse e richiedono un coordinamento tra i servizi indicati sopra.

Jarvis è un framework applicativo completamente accelerato per la creazione di servizi ai di conversazione

multimodale che utilizzano una pipeline di deep learning end-to-end. Il framework Jarvis include modelli di ai conversazionali preformati, strumenti e servizi end-to-end ottimizzati per le attività vocali, di visione e NLU. Oltre ai servizi di intelligenza artificiale, Jarvis ti consente di unire contemporaneamente vision, audio e altri input dei sensori per offrire funzionalità come conversazioni multi-utente e multi-contesto in applicazioni come assistenti virtuali, diarizzazione multiutente e assistenti di call center.

NVIDIA NEMO

"NVIDIA NEMO" È un toolkit Python open-source per la creazione, la formazione e la messa a punto di modelli di ai conversazionali allo stato dell'arte con accelerazione GPU utilizzando interfacce di programmazione applicativa (API) di facile utilizzo. NEMO esegue calcoli misti di precisione utilizzando core Tensor in GPU NVIDIA e può scalare facilmente fino a più GPU per offrire le migliori performance di training possibili. NEMO viene utilizzato per creare modelli per applicazioni ASR, NLP e TTS in tempo reale, come trascrizioni di videochiamate, assistenti video intelligenti e supporto automatizzato di call center in diversi mercati verticali del settore, tra cui settore sanitario, finanziario, retail e telecomunicazioni.

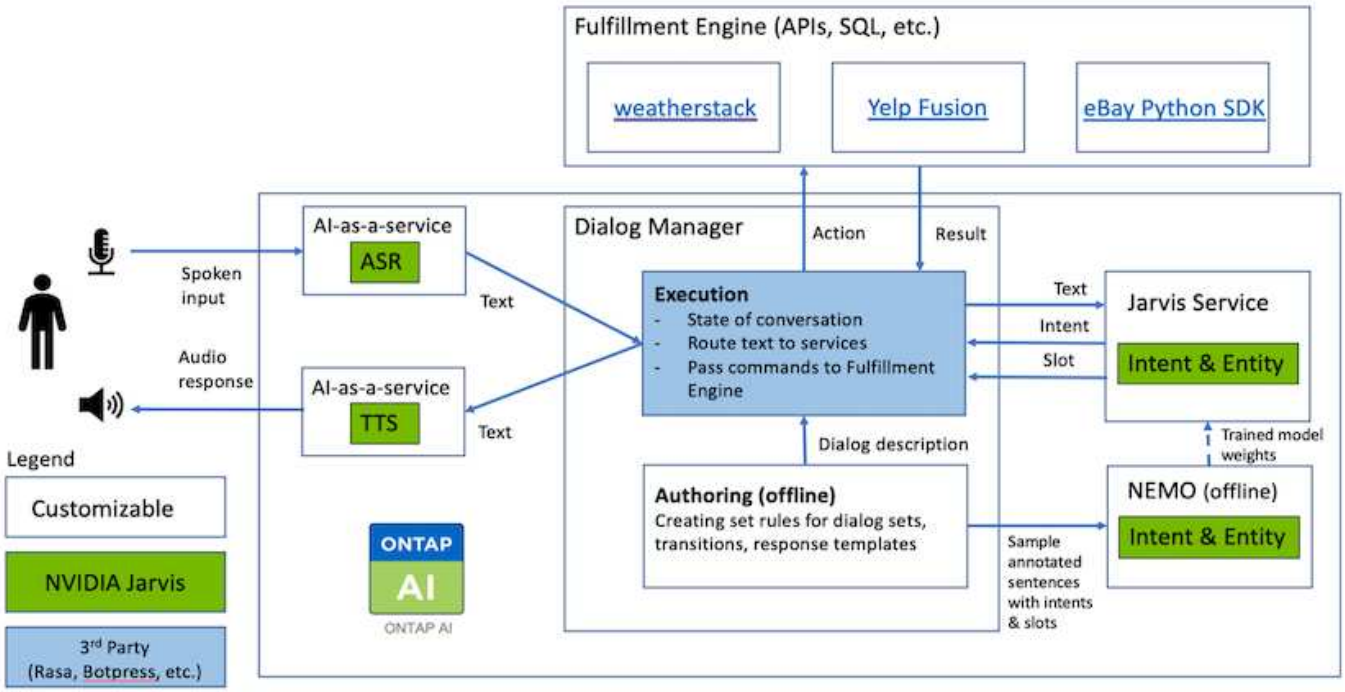
Abbiamo utilizzato NEMO per formare modelli che riconoscono intenti complessi dalle domande degli utenti nella cronologia delle conversazioni archiviate. Questo training estende le funzionalità dell'assistente virtuale al dettaglio oltre il supporto fornito da Jarvis.

Riepilogo dei casi di utilizzo al dettaglio

Utilizzando NVIDIA Jarvis, abbiamo creato un assistente virtuale al dettaglio che accetta l'input vocale o di testo e risponde a domande relative a meteo, punti di interesse e prezzi di inventario. Il sistema ai conversazionale è in grado di ricordare il flusso di conversazione, ad esempio, porre una domanda di follow-up se l'utente non specifica la posizione per il meteo o i punti di interesse. Il sistema riconosce anche entità complesse come "cibo tailandese" o "memoria per laptop". Comprende domande di linguaggio naturale come "pioverà la prossima settimana a Los Angeles?" Una dimostrazione dell'assistente virtuale per la vendita al dettaglio è disponibile in "Personalizza gli stati e i flussi per i casi d'utilizzo retail".

Tecnologia della soluzione

La figura seguente illustra l'architettura di sistema ai conversazionale proposta. È possibile interagire con il sistema sia con il segnale vocale che con l'immissione di testo. Se viene rilevato un input vocale, Jarvis ai-as-service (AIaaS) esegue ASR per produrre testo per Dialog Manager. Dialog Manager memorizza gli stati di conversazione, indirizza il testo ai servizi corrispondenti e passa i comandi al motore di adempimento. Jarvis NLP Service prende il testo, riconosce intenti ed entità e restituisce tali intenti e slot di entità a Dialog Manager, che invia quindi Action al motore di adempimento. Fulfillment Engine è costituito da API di terze parti o database SQL che rispondono alle query degli utenti. Dopo aver ricevuto il risultato da Fulfillment Engine, Dialog Manager indirizza il testo a Jarvis TTS AIaaS per produrre una risposta audio per l'utente finale. Possiamo archiviare la cronologia delle conversazioni, annotare frasi con intenti e slot per il training NEMO in modo che il servizio NLP migliori man mano che un maggior numero di utenti interagisce con il sistema.



Requisiti hardware

Questa soluzione è stata validata utilizzando una stazione DGX e un sistema storage AFF A220. Jarvis richiede una GPU T4 o V100 per eseguire calcoli di rete neurali profondi.

La seguente tabella elenca i componenti hardware necessari per implementare la soluzione come testata.

Hardware	Quantità
GPU T4 O V100	1
Stazione NVIDIA DGX	1

Requisiti software

La seguente tabella elenca i componenti software necessari per implementare la soluzione come testata.

Software	Versione o altre informazioni
Software per la gestione dei dati NetApp ONTAP	9.6
Firmware dello switch Cisco NX-OS	7.0(3)I6(1)
SISTEMA OPERATIVO NVIDIA DGX	4.0.4 - Ubuntu 18.04 LTS
NVIDIA Jarvis Framework	EA v0.2
NVIDIA NEMO	nvcr.io/nvidia/nemo:v0.10
Piattaforma container Docker	18.06.1-ce [e68fc7a]

Panoramica

In questa sezione vengono fornite informazioni dettagliate sull'implementazione di Virtual Retail Assistant.

Implementazione di Jarvis

Puoi iscriverti a. "Programma Jarvis Early Access" Per accedere ai container Jarvis su NVIDIA GPU Cloud (NGC). Dopo aver ricevuto le credenziali da NVIDIA, puoi implementare Jarvis seguendo questa procedura:

1. Accedi a NGC.
2. Imposta la tua organizzazione su NGC: `ea-2-jarvis`.
3. Individuare le risorse Jarvis EA v0.2: I container Jarvis sono in `Private Registry` > `Organization Containers`.
4. Selezionare Jarvis: Selezionare `Model Scripts` e fare clic su `Jarvis Quick Start`
5. Verificare che tutte le risorse funzionino correttamente.
6. Trova la documentazione per creare le tue applicazioni: I PDF sono disponibili in `Model Scripts` > `Jarvis Documentation` > `File Browser`.

Personalizza gli stati e i flussi per i casi d'utilizzo retail

È possibile personalizzare gli stati e i flussi di Dialog Manager in base ai casi di utilizzo specifici. Nel nostro esempio di vendita al dettaglio, abbiamo i seguenti quattro file yaml per indirizzare la conversazione in base a diversi intenti.

Se il seguente elenco di nomi di file e la descrizione di ciascun file:

- `main_flow.yml`: Definisce i flussi e gli stati principali della conversazione e indirizza il flusso agli altri tre file yaml, se necessario.
- `retail_flow.yml`: Contiene stati relativi a domande al dettaglio o punti di interesse. Il sistema fornisce le informazioni del negozio più vicino o il prezzo di un dato articolo.
- `weather_flow.yml`: Contiene gli stati relativi alle domande sul meteo. Se non è possibile determinare la posizione, il sistema pone una domanda di follow-up per chiarire.
- `error_flow.yml`: Gestisce i casi in cui gli intenti dell'utente non rientrano nei tre file yaml precedenti. Dopo aver visualizzato un messaggio di errore, il sistema torna ad accettare le domande dell'utente.le sezioni seguenti contengono le definizioni dettagliate per questi file 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
```

Connettersi alle API di terze parti come motore di adempimento

Abbiamo collegato le seguenti API di terze parti come motore di adempimento per rispondere alle domande:

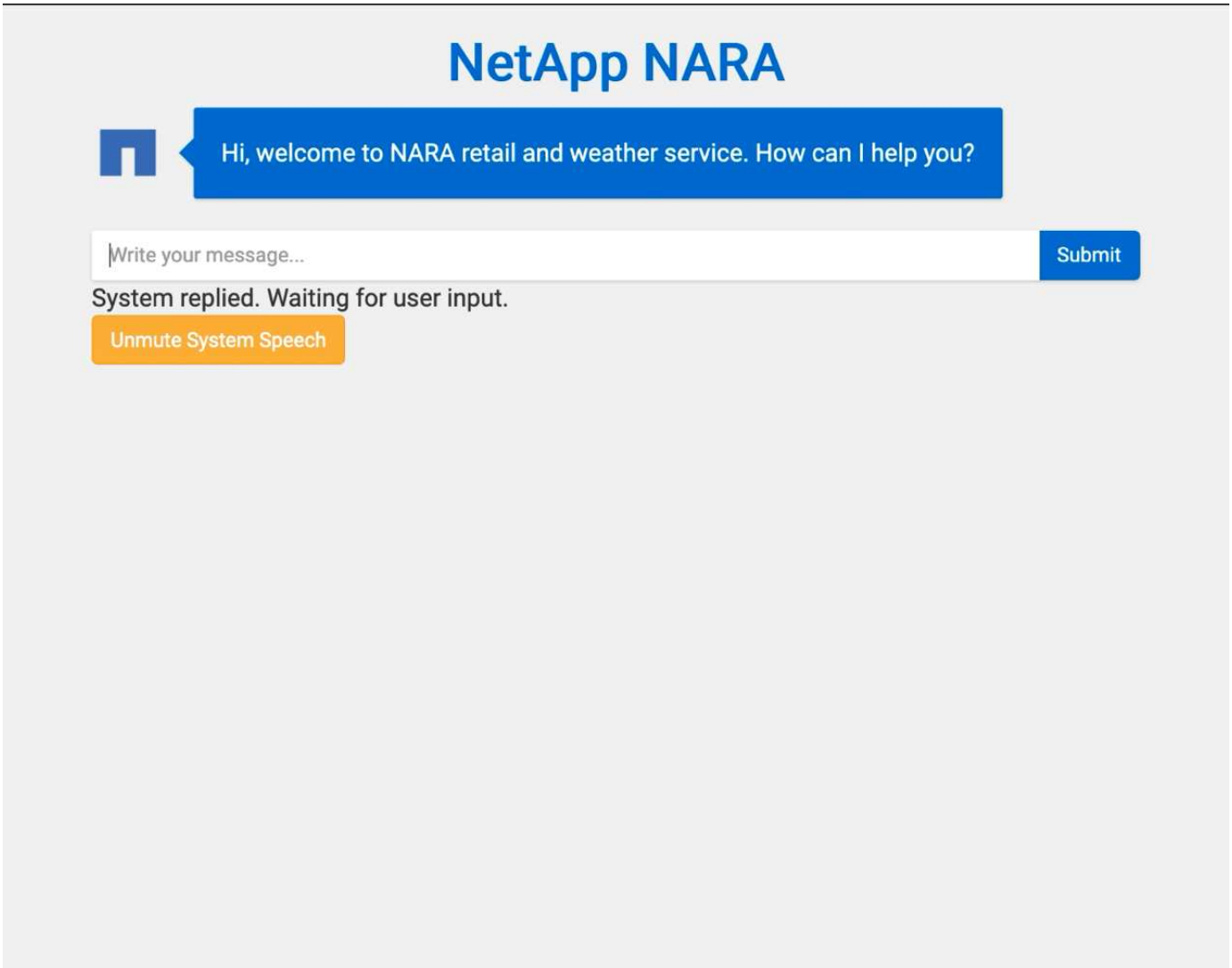
- "API di WeatherStack": restituisce meteo, temperatura, pioggia e neve in una determinata posizione.
- "API Fusion di Yelp": restituisce le informazioni del negozio più vicino in una determinata posizione.
- "SDK di eBay Python": restituisce il prezzo di un dato articolo.

Dimostrazione di NetApp Retail Assistant

Abbiamo registrato un video dimostrativo di NetApp Retail Assistant (NARA).

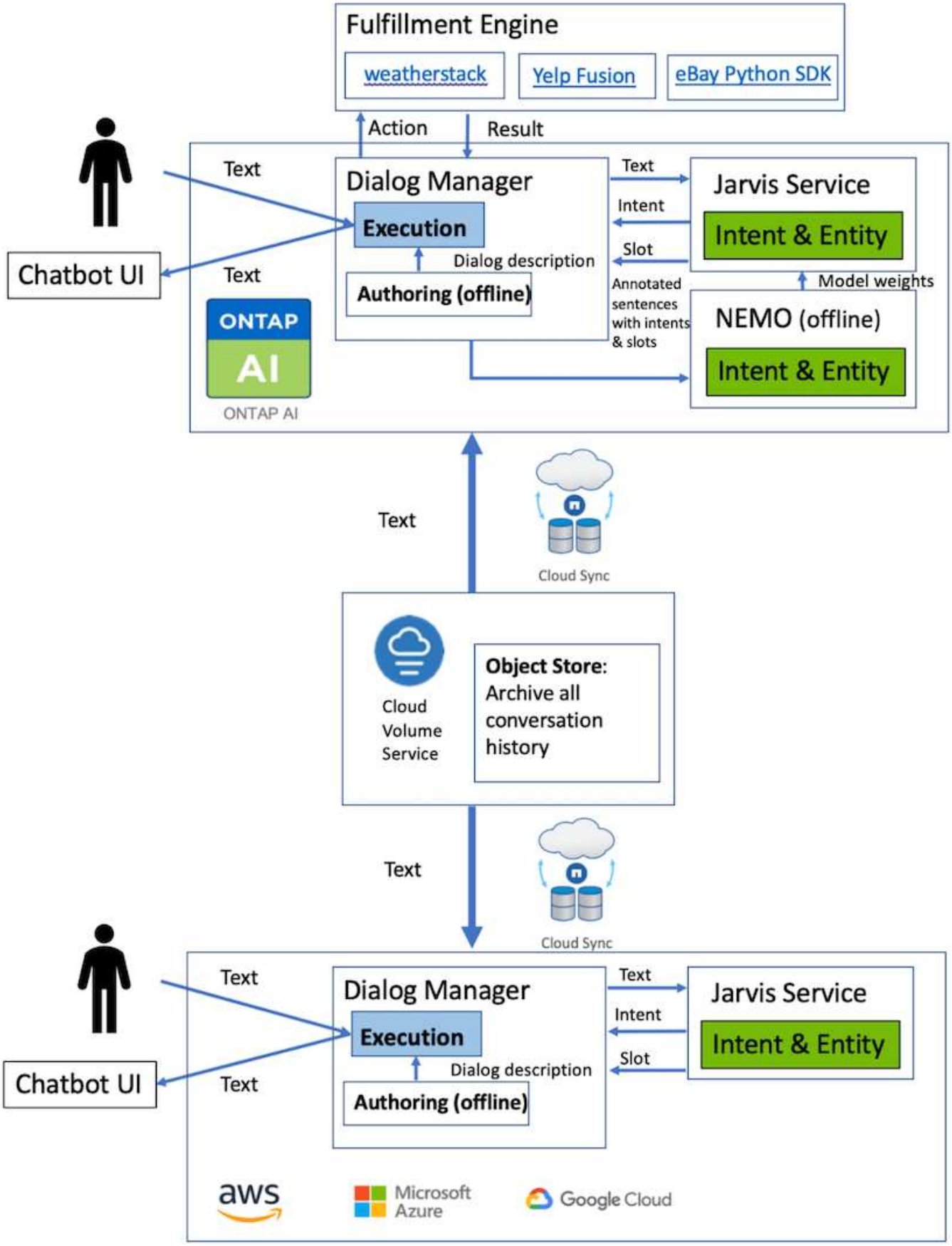
Video dimostrativo DI NARA

Video dimostrativo DI NARA



Utilizza la copia e sincronizzazione di NetApp BlueXP per archiviare la cronologia delle conversazioni

Scaricando la cronologia delle conversazioni in un file CSV una volta al giorno, possiamo quindi sfruttare BlueXP Copy e Sync per scaricare i file di log nello storage locale. La figura seguente mostra l'architettura di avere implementato Jarvis on-premise e nei cloud pubblici, mentre utilizza BlueXP Copy e Sync per inviare la cronologia delle conversazioni per il training NEMO. I dettagli del training NEMO sono disponibili nella sezione "Espandi i modelli di intento utilizzando 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

Espandi i modelli di intento utilizzando NEMO Training

NVIDIA NEMO è un toolkit creato da NVIDIA per la creazione di applicazioni ai conversazionali. Questo toolkit include raccolte di moduli pre-formati per ASR, NLP e TTS, che consentono a ricercatori e data scientist di comporre facilmente architetture di rete neurali complesse e di concentrarsi maggiormente sulla progettazione delle proprie applicazioni.

Come illustrato nell'esempio precedente, NARA può gestire solo un tipo limitato di domanda. Questo perché il modello di NLP pre-addestrato si allena solo su questi tipi di domande. Se vogliamo consentire A NARA di gestire una gamma più ampia di domande, dobbiamo rielaborare il sistema con i nostri set di dati. In questo caso, dimostreremo come possiamo utilizzare NEMO per estendere il modello NLP in modo da soddisfare i requisiti. Iniziamo convertendo il log raccolto da NARA nel formato NEMO, quindi ci alleniamo con il set di dati per migliorare il modello NLP.

Modello

Il nostro obiettivo è consentire A NARA di ordinare gli elementi in base alle preferenze dell'utente. Ad esempio, potremmo chiedere A NARA di suggerire il ristorante di sushi più classificato o di cercare I jeans CON il prezzo più basso. A tal fine, utilizziamo il modello di rilevamento degli intenti e di riempimento degli slot fornito in NEMO come modello di training. Questo modello consente A NARA di comprendere l'intento della ricerca delle preferenze.

Preparazione dei dati

Per formare il modello, raccogliamo il dataset per questo tipo di domanda e lo convertiamo nel formato NEMO. Qui sono elencati i file utilizzati per la formazione del modello.

dict.intents.csv

Questo file elenca tutti gli intenti che vogliamo che NEMO comprenda. In questo caso, abbiamo due intenti primari e un solo intento utilizzato per classificare le domande che non si inseriscono in nessuno degli intenti primari.

```
price_check
find_the_store
unknown
```

dict.slots.csv

Questo file elenca tutti gli slot che possiamo etichettare sulle nostre domande di training.

```
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.sv

Questo è il set di dati di training principale. Ogni riga inizia con la domanda che segue l'elenco delle categorie di intento nel file dict.intent.csv. L'etichetta viene enumerata a partire da zero.

train_slot.sv

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

Formare il modello

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

Quindi, viene utilizzato il seguente comando per avviare il container. In questo comando, limitiamo il container a utilizzare una singola GPU (ID GPU = 1), poiché si tratta di un esercizio di formazione leggero. Inoltre, mappiamo la nostra area di lavoro locale /Workspace/nemo/ nella cartella all'interno di container /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
```

All'interno del container, se si desidera partire dal modello BERT originale pre-addestrato, è possibile utilizzare il seguente comando per avviare la procedura di training. data_dir è l'argomento per impostare il percorso dei dati di training. work_dir consente di configurare la posizione in cui si desidera memorizzare i file del punto di verifica.

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

Se abbiamo nuovi set di dati di training e vogliamo migliorare il modello precedente, possiamo utilizzare il seguente comando per continuare dal punto in cui ci siamo fermati. checkpoint_dir porta il percorso alla cartella checkpoint precedente.

```
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
```

Deduzione del modello

Dobbiamo convalidare le performance del modello formatosi dopo un certo numero di epoche. Il seguente comando consente di eseguire il test della query uno per uno. Ad esempio, in questo comando, si desidera verificare se il modello è in grado di identificare correttamente l'intenzione della query `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
```

Di seguito viene riportato l'output dell'inferenza. Nell'output, possiamo vedere che il nostro modello addestrato può prevedere correttamente l'intenzione find_the_store e restituire le parole chiave a cui siamo interessati. Con queste parole chiave, consentiamo A NARA di cercare ciò che gli utenti desiderano e di effettuare una ricerca più 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
```

Conclusione

Un vero e proprio sistema di ai conversa si impegna in un dialogo umano, comprende il contesto e fornisce risposte intelligenti. Tali modelli di ai sono spesso enormi e altamente complessi. Con le GPU NVIDIA e lo storage NetApp, è possibile formare e ottimizzare

modelli di linguaggio all'avanguardia per eseguire rapidamente l'inferenza. Si tratta di un importante passo avanti verso la fine del compromesso tra un modello di ai veloce e uno grande e complesso. I modelli di comprensione del linguaggio ottimizzati per la GPU possono essere integrati nelle applicazioni di ai per settori come l'assistenza sanitaria, la vendita al dettaglio e i servizi finanziari, alimentando assistenti vocali digitali avanzati in altoparlanti intelligenti e linee di assistenza clienti. Questi sistemi di ai convergenti di alta qualità consentono alle aziende di tutti i mercati verticali di fornire servizi personalizzati precedentemente irraggiungibili quando si impegnano con i clienti.

Jarvis consente l'implementazione di casi di utilizzo come assistenti virtuali, avatar digitali, Fusion del sensore multimodale (CV fuso con ASR/NLP/TTS) o qualsiasi caso di utilizzo autonomo ASR/NLP/TTS/CV, ad esempio la trascrizione. Abbiamo creato un assistente virtuale al dettaglio in grado di rispondere a domande relative a meteo, punti di interesse e prezzi dell'inventario. Abbiamo anche dimostrato come migliorare le funzionalità di comprensione del linguaggio naturale del sistema ai conversazionale archiviando la cronologia delle conversazioni utilizzando BlueXP Copy and Sync e formando i modelli NEMO sui nuovi dati.

Ringraziamenti

Gli autori riconoscono con gratitudine i contributi che sono stati apportati a questo white paper dai nostri stimati colleghi di NVIDIA: Davide Onofrio, Alex Qi, Sicong Ji, Marty Jain e Robert Sohigian. Gli autori desiderano inoltre ringraziare i principali membri del team NetApp: Santosh Rao, David Arnette, Michael Oglesby, Brent Davis, Andy Sayare, Erik Mulder e Mike McNamara.

Il nostro sincero apprezzamento e ringraziamento va a tutti questi individui, che hanno fornito informazioni e competenze che hanno contribuito enormemente alla creazione di questo documento.

Dove trovare ulteriori informazioni

Per ulteriori informazioni sulle informazioni descritte in questo documento, consultare le seguenti risorse:

- NVIDIA DGX Station, V100 GPU, GPU Cloud
  - Stazione NVIDIA DGXhttps://www.nvidia.com/en-us/data-center/dgx-station/["https://www.nvidia.com/en-us/data-center/dgx-station/"^]
  - NVIDIA V100 Tensor Core GPUhttps://www.nvidia.com/en-us/data-center/tesla-v100/["https://www.nvidia.com/en-us/data-center/tesla-v100/"^]
  - NVIDIA NGChttps://www.nvidia.com/en-us/gpu-cloud/["https://www.nvidia.com/en-us/gpu-cloud/"^]
- Framework multimodale NVIDIA Jarvis
  - NVIDIA Jarvishttps://developer.nvidia.com/nvidia-jarvis["https://developer.nvidia.com/nvidia-jarvis"^]
  - Accesso anticipato a NVIDIA Jarvishttps://developer.nvidia.com/nvidia-jarvis-early-access["https://developer.nvidia.com/nvidia-jarvis-early-access"^]
- NVIDIA NEMO
  - NVIDIA NEMOhttps://developer.nvidia.com/nvidia-nemo["https://developer.nvidia.com/nvidia-nemo"^]
  - Guida per sviluppatorihttps://nvidia.github.io/NeMo/["https://nvidia.github.io/NeMo/"^]

- Sistemi NetApp AFF
  - Scheda informativa su NetApp AFF Serie A.https://www.netapp.com/us/media/ds-3582.pdf["https://www.netapp.com/us/media/ds-3582.pdf"^]
  - NetApp Flash Advantage per All Flash FAShttps://www.netapp.com/us/media/ds-3733.pdf["https://www.netapp.com/us/media/ds-3733.pdf"^]
  - Raccolta di informazioni su ONTAP 9http://mysupport.netapp.com/documentation/productlibrary/index.html?productID=62286["http://mysupport.netapp.com/documentation/productlibrary/index.html?productID=62286"^]
  - Report tecnico NetApp ONTAP FlexGroup Volumeshttps://www.netapp.com/us/media/tr-4557.pdf["https://www.netapp.com/us/media/tr-4557.pdf"^]
- NetApp ONTAP ai
  - Guida alla progettazione di reti ONTAP ai con DGX-1 e Ciscohttps://www.netapp.com/us/media/nva-1121-design.pdf["https://www.netapp.com/us/media/nva-1121-design.pdf"^]
  - Guida all'implementazione di ONTAP ai con DGX-1 e Cisco Networkinghttps://www.netapp.com/us/media/nva-1121-deploy.pdf["https://www.netapp.com/us/media/nva-1121-deploy.pdf"^]
  - Guida alla progettazione di reti ONTAP ai con DGX-1 e Mellanoxhttp://www.netapp.com/us/media/nva-1138-design.pdf["http://www.netapp.com/us/media/nva-1138-design.pdf"^]
  - Guida alla progettazione di ONTAP ai con DGX-2https://www.netapp.com/us/media/nva-1135-design.pdf["https://www.netapp.com/us/media/nva-1135-design.pdf"^]