Umgangssprachlich AI mit NVIDIA

NetApp Solutions

NetApp
April 19, 2024

This PDF was generated from https://docs.netapp.com/de-de/netapp-solutions/ai/cainvidia_solution_overview.html on April 19, 2024. Always check docs.netapp.com for the latest.

Inhalt

WP-7328: NetApp Convergentional AI mit NVIDIA Jarvis . . . 1
- Lösungsüberblick . . . 1
- Lösungstechnologie . . . 3
- Überblick . . . 5
- Schlussfolgerung . . . 19
- Danksagungen . . . 20
- Weitere Informationen . . . 20

WP-7328: NetApp Convergentional AI mit NVIDIA Jarvis

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

Die NVIDIA DGX-Systemfamilie besteht aus den weltweit ersten integrierten Systemen für künstliche Intelligenz (KI), die speziell für Enterprise-KI entwickelt wurden. NetApp AFF Storage-Systeme bieten extrem hohe Performance und branchenführende Funktionen für das Hybrid-Cloud-Datenmanagement. NetApp und NVIDIA haben gemeinsam die NetApp ONTAP AI Referenzarchitektur entwickelt, eine sofort einsatzbereite Lösung für KI- und ML-Workloads (Machine Learning) mit Performance, Zuverlässigkeit und Support der Enterprise-Klasse.

Dieses Whitepaper gibt Kunden eine Orientierungshilfe beim Aufbau konversationaler KI-Systeme zur Unterstützung verschiedener Anwendungsfälle in verschiedenen Branchen. Es enthält Informationen über die Bereitstellung des Systems mit NVIDIA Jarvis. Die Tests wurden mit einer NVIDIA DGX Station und einem NetApp AFF A220 Storage-System durchgeführt.

Die Zielgruppe für die Lösung umfasst die folgenden Gruppen:

- Enterprise-Architekten, die Lösungen für die Entwicklung von KI-Modellen und -Software für umgangssprachlich KI-Anwendungsfälle wie einen virtuellen Einzelhandelsassistent entwerfen
- Data Scientists, die nach effizienten Möglichkeiten suchen, Ziele bei der Sprachmodellierung zu erreichen
- Dateningenieure, die für die Pflege und Verarbeitung von Textdaten wie Kundenfragen und Dialogtranskripte verantwortlich sind
- Führungskräfte und IT-Entscheidungsträger, die daran interessiert sind, die konvergente KI-Erfahrung zu verändern und mit KI-Initiativen die kürzeste Markteinführungszeit zu erreichen

Lösungsüberblick

NetApp ONTAP AI und BlueXP Kopier- und Synchronisierungsfunktion

Die NetApp ONTAP KI-Architektur mit NVIDIA DGX -Systemen und Cloud-vernetzten NetApp Storage-Systemen wurde von NetApp und NVIDIA entwickelt und verifiziert. Diese Referenzarchitektur bietet IT-Abteilungen folgende Vorteile:

- Sie vereinfacht das Design
- Computing und Storage können unabhängig voneinander skaliert werden
- Kunden können mit einer kleinen Konfiguration starten und nahtlos skalieren
- Bietet eine Reihe von Storage-Optionen für verschiedene Performance- und KostenpunkteNetApp ONTAP AI integriert DGX-Systeme und NetApp AFF A220 Storage-Systeme nahtlos in hochmoderne Netzwerke. Mithilfe von NetApp ONTAP AI und DGX-Systemen können Komplexität und Unsicherheiten bei der Systemaufsetzung beseitigt werden, was den Einsatz von KI-Projekten vereinfacht. Kunden können mit einer kleinen Installation starten und ihre Systeme unterbrechungsfrei erweitern. Gleichzeitig erhalten sie intelligente Datenmanagement-Funktionen, mit denen sich Daten zwischen Datenaufnahme, zentraler Datenplattform und Cloud frei verschieben lassen.

Mit NetApp BlueXP Kopier- und Synchronisierungsfunktion können Sie Daten einfach über verschiedene

Protokolle verschieben, egal ob sich zwischen zwei NFS-Freigaben, zwei CIFS-Freigaben oder zwischen einer Dateifreigabe und Amazon S3, Amazon Elastic File System (EFS) oder Azure Blob Storage befindet. Durch die aktiv/aktiv-Vorgänge ist gleichzeitiges Arbeiten an Daten am Quell- und am Zielort möglich, während die Datenänderungen bei Bedarf inkrementell synchronisiert werden. BlueXP Copy and Sync ermöglicht Ihnen das Verschieben und inkrementelle Synchronisieren von Daten zwischen beliebigen Quell- und Zielsystemen – lokal oder in der Cloud – und eröffnet Ihnen eine Vielzahl neuer Möglichkeiten für die Datennutzung. Das Migrieren von Daten zwischen On-Premises-Systemen, Cloud-On-Boarding und Cloud-Migration oder Collaboration und Datenanalysen ist leicht möglich. Die Abbildung unten zeigt die verfügbaren Quellen und Ziele.

In Konversations-KI-Systemen können Entwickler BlueXP Copy und Sync nutzen, um den Gesprächsverlauf von der Cloud in Datacentern zu archivieren und damit das Offline-Training von NLP-Modellen (Natural Language Processing) zu ermöglichen. Durch Training von Modellen, um mehr Absichten zu erkennen, wird das Conversational KI-System besser gerüstet sein, um komplexere Fragen von den Endbenutzern zu managen.

NVIDIA Jarvis Multimodal Framework

"NVIDIA Jarvis" Ist ein durchgängiges Framework für die Erstellung umgangssprachlicher KI-Services. Es umfasst die folgenden GPU-optimierten Services:

- Automatische Spracherkennung (ASR)
- Verständnis natürlicher Sprachen (NLU)
- Integration in domänenspezifische Fulfillment Services
- Text-to-Speech (TTS)
- Computer Vision (CV)Jarvis-basierte Services nutzen moderne Deep-Learning-Modelle, um der komplexen und anspruchsvollen Aufgabe von Echtzeit-konvergierter KI gerecht zu werden. Um natürliche Echtzeit-Interaktionen mit einem Endbenutzer zu ermöglichen, müssen die Modelle die Berechnungen in weniger als 300 Millisekunden durchführen. Natürliche Interaktionen sind anspruchsvoll und erfordern eine multimodale sensorische Integration. Modell-Pipelines sind auch komplex und erfordern Koordination über die oben genannten Services.

Jarvis ist ein vollständig beschleunigtes Applikations-Framework für die Erstellung multimodaler, umgangssprachlicher KI-Services, die eine End-to-End-Deep-Learning-Pipeline nutzen. Das Jarvis-Framework umfasst vortrainierte konvergente KI-Modelle, Tools und optimierte End-to-End-Services für sprach-, Vision- und NLU-Aufgaben. Neben den KI-Services ermöglicht Jarvis die gleichzeitige Sicherung von Bild-, Audio- und anderen Sensoreingängen, um Funktionen wie Multi-User-, Multi-Context-Gespräche in Anwendungen wie virtuellen Assistenten, Multi-User-Diarization und Callcenter-Assistenten bereitzustellen.

NVIDIA Nemo

"NVIDIA Nemo" Es ist ein Open-Source-Python-Toolkit für die Erstellung, das Training und die Feinabstimmung von GPU-beschleunigten, hochmodernen KI-Modellen mit benutzerfreundlichen APIs (Application Programming Interfaces). Nemo arbeitet mit Tensor Cores in NVIDIA GPUs und kann bis auf mehrere GPUs skaliert werden, um die höchstmögliche Trainings-Performance zu erzielen. Nemo wird zur Erstellung von Modellen für ASR-, NLP- und TTS-Anwendungen in Echtzeit verwendet, wie z. B. Videoanrufe-Transkriptionen, intelligente Videoassistenten und automatisierte Unterstützung von Callcenter in verschiedenen Branchen, einschließlich Gesundheitswesen, Finanzen, Einzelhandel und Telekommunikation.

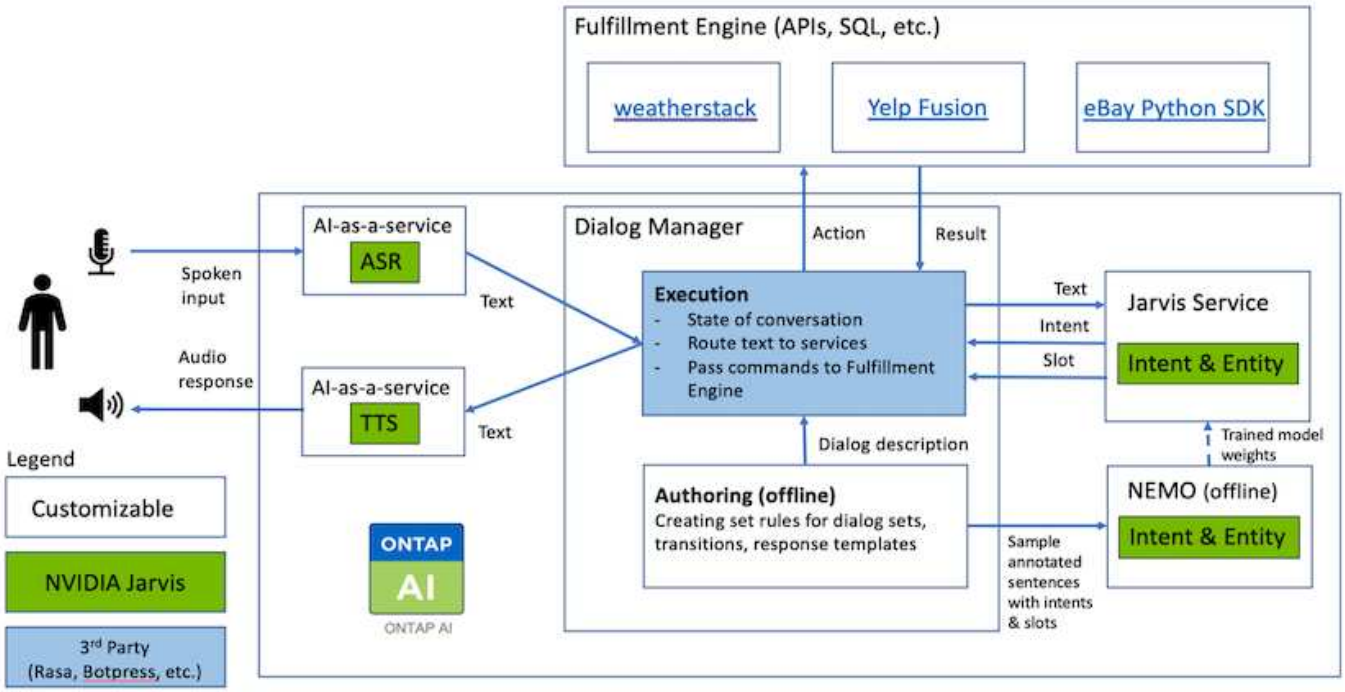
Mit Nemo trainierte man Modelle, die komplexe Absichten aus Benutzerfragen in der Vergangenheit archivierter Gespräche erkennen. Diese Schulung erweitert die Fähigkeiten des virtuellen Einzelhandels-Assistenten über das hinaus, was Jarvis als bereitgestellt unterstützt.

Anwendungsbeispiel Im Einzelhandel – Zusammenfassung

Mithilfe von NVIDIA Jarvis haben wir einen virtuellen Retail-Assistenten entwickelt, der sprach- oder Texteingaben akzeptiert und Fragen zu Wetter, Points of Interest und Bestandspreisen beantwortet. Das Conversational AI-System kann sich an den Gesprächsablauf erinnern, beispielsweise eine weitere Frage stellen, wenn der Benutzer nicht den Standort für Wetter oder Points of Interest angeben kann. Das System erkennt auch komplexe Einheiten wie „Thai Food“ oder „Laptop-Speicher“. Es versteht natürliche Sprachfragen wie „regnet es nächste Woche in Los Angeles?“ Eine Demonstration des virtuellen Assistenten für den Einzelhandel finden Sie in "Passen Sie die Staaten und Abläufe für den Einzelhandel an".

Lösungstechnologie

Die folgende Abbildung zeigt die vorgeschlagene umgangssprachlichen KI-Systemarchitektur. Sie können mit dem System entweder mit Sprachsignal- oder Texteingabe interagieren. Wenn gesprochene Eingabe erkannt wird, führt Jarvis AI-as-Service (AIAAS) ASR aus, um Text für Dialog Manager zu erstellen. Dialog Manager speichert die Zustände der Konversation, leitet Text zu den entsprechenden Diensten weiter und leitet Befehle an Fulfillment Engine weiter. Jarvis NLP Service nimmt Text ein, erkennt Intents und Entities und gibt diese Intents und Entity Slots zurück an Dialog Manager, der dann Aktion an Fulfillment Engine sendet. Fulfillment Engine besteht aus Drittanbieter-APIs oder SQL-Datenbanken, die Benutzeranfragen beantworten. Nach Erhalt der Ergebnisse von Fulfillment Engine leitet Dialogue Manager Text an Jarvis TTS AIAaS weiter, um eine Audioantwort für den Endbenutzer zu erstellen. Wir können Gesprächsverlauf archivieren, Sätze mit Intents und Slots für Nemo-Schulungen annotieren, so dass der NLP-Service verbessert wird, wenn mehr Benutzer mit dem System interagieren.



Hardwareanforderungen

Diese Lösung wurde mit einer DGX-Station und einem AFF A220 Storage-System validiert. Jarvis benötigt entweder eine T4 oder V100 GPU zur Durchführung einer tiefen neuronalen Netzwerkcompuation.

In der folgenden Tabelle werden die Hardwarekomponenten aufgeführt, die für die Implementierung der getesteten Lösung erforderlich sind.

Trennt	Menge
T4 ODER V100 GPU	1
NVIDIA DGX Station	1

Softwareanforderungen

In der folgenden Tabelle werden die Softwarekomponenten aufgeführt, die für die Implementierung der Lösung erforderlich sind.

Software	Version oder sonstige Informationen
NetApp ONTAP Datenmanagement-Software	9.6
Cisco NX-OS Switch-Firmware	7.0(3)I6(1)
NVIDIA DGX-BETRIEBSSYSTEM	4.0.4 – Ubuntu 18.04 LTS
NVIDIA Jarvis Framework	EA v0.2
NVIDIA Nemo	Nvcr.io/nvidia/nemo:v0.10
Docker-Container-Plattform	18.06.1-ce [e68fc7a]

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

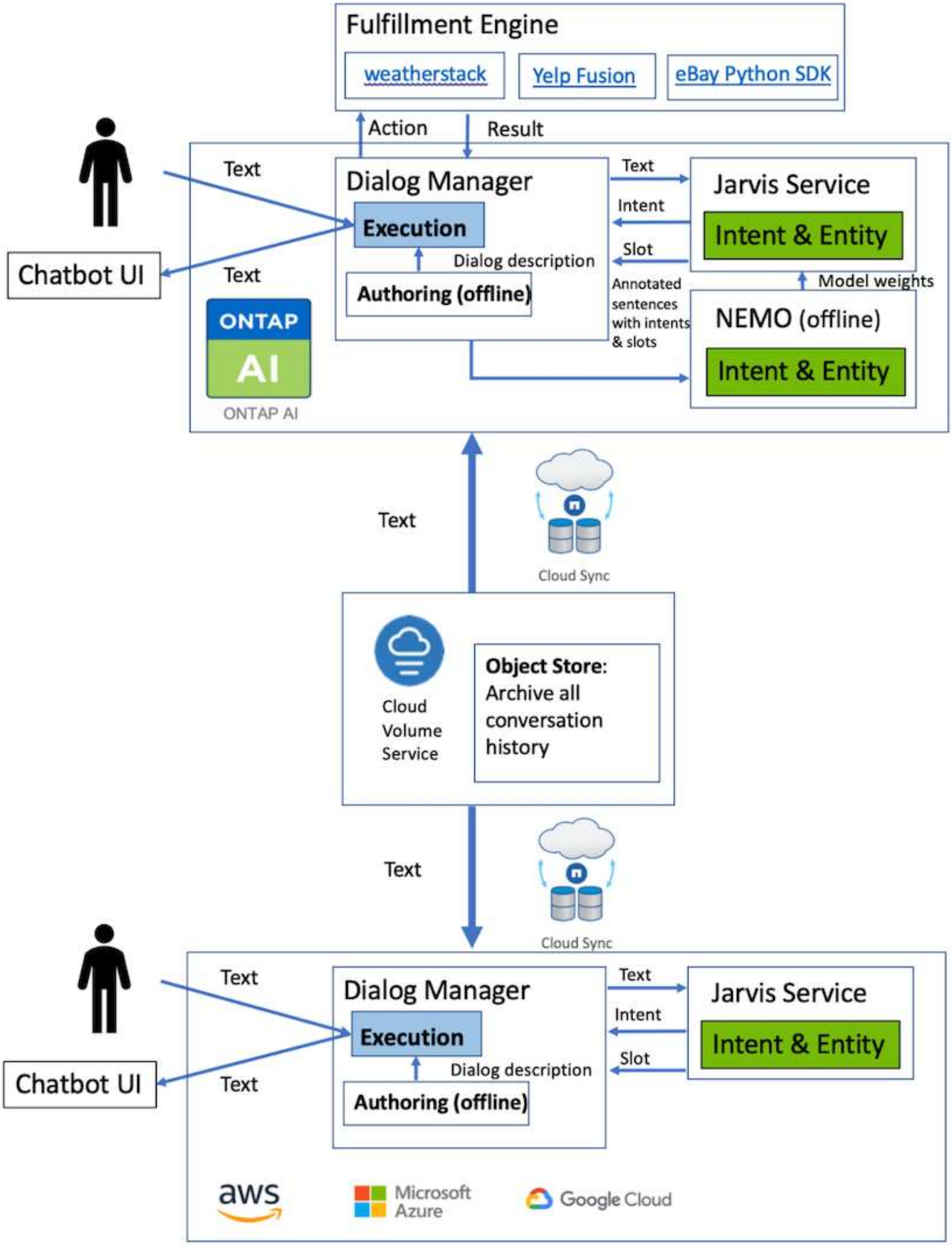
Wir haben ein Demovideo zu dem NetApp Retail Assistant (NARA) aufgenommen.

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
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

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
```

Schlussfolgerung

Ein echtes Conversational KI-System engagiert sich in einem menschlichen Dialog, versteht Kontext und bietet intelligente Antworten. Solche KI-Modelle sind oft riesig und sehr komplex. Mit NVIDIA GPUs und NetApp Storage können große, moderne

Sprachmodelle trainiert und optimiert werden, um Inferenz schnell auszuführen. Dies ist ein wichtiger Schritt in Richtung einer Beendigung des Handels zwischen einem KI-Modell, das schnell ist, als ein großes und komplexes Modell. GPU-optimierte Modelle zur Sprachverständnisfunktion können für Branchen wie Gesundheitswesen, Einzelhandel und Finanzdienstleistungen in KI-Applikationen integriert werden und unterstützen digitale Sprachassistenten in intelligenten Lautsprechern und Kundenservices. Diese hochwertigen konvergenten KI-Systeme ermöglichen Unternehmen in verschiedenen Branchen, ihren Kundeneinbezug als bisher unerreichbare, personalisierte Services bereitzustellen.

Jarvis ermöglicht die Bereitstellung von Anwendungsbeispielen wie virtuellen Assistenten, digitalen Avataren, multimodalen Sensor Fusion (CV fusioned with ASR/NLP/TTS) oder jedem ASR/NLP/TTS/CV-Einzelanwendungsfall, wie z.B. Transkription. Wir haben einen virtuellen Einzelhandelsassistent aufgebaut, der Fragen zu Wetter, Points of Interest und Bestandspreisen beantworten kann. Wir haben auch gezeigt, wie die Fähigkeiten des Konversations-KI-Systems zum Verstehen der natürlichen Sprache verbessert werden können. Dazu haben wir den Gesprächsverlauf mithilfe von BlueXP Copy und Sync archiviert und Nemo-Modelle auf neuen Daten trainiert.

Danksagungen

Die Autoren bedanken sich herzlich für die Beiträge, die zu diesem White Paper von unseren geschätzten Kollegen von NVIDIA gemacht wurden: Davide Onofrio, Alex Qi, Sicong Ji, Marty Jain und Robert Sohigian. Bedanken möchten sie sich auch für die Beiträge der zentralen NetApp Teammitglieder Santosh Rao, David Arnette, Michael Oglesby, Brent Davis, Andy Sayare, Erik Mulder und Mike McNamara.

Unsere aufrichtige Wertschätzung und unser Dank richten sich an jeden Einzelnen, der mit seinen Einblicken und seinem Know-how einen wichtigen Beitrag zu diesem Dokument geleistet hat.

Weitere Informationen

Folgende Ressourcen enthalten ausführlichere Informationen zu den Angaben in diesem Dokument:

- NVIDIA DGX Station, V100 GPU, GPU Cloud
  - NVIDIA DGX Stationhttps://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/"^]
- NVIDIA Jarvis Multimodal Framework
  - NVIDIA Jarvishttps://developer.nvidia.com/nvidia-jarvis["https://developer.nvidia.com/nvidia-jarvis"^]
  - NVIDIA Jarvis Early Accesshttps://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"^]

    - Entwicklerhandbuchhttps://nvidia.github.io/NeMo/["https://nvidia.github.io/NeMo/"^]
- NetApp AFF Systeme
    - NetApp AFF A-Series – Datenblatthttps://www.netapp.com/us/media/ds-3582.pdf["https://www.netapp.com/us/media/ds-3582.pdf"^]
    - NetApp Flash Advantage für All Flash FAShttps://www.netapp.com/us/media/ds-3733.pdf["https://www.netapp.com/us/media/ds-3733.pdf"^]
    - ONTAP 9 Informationsbibliothekhttp://mysupport.netapp.com/documentation/productlibrary/index.html?productID=62286["http://mysupport.netapp.com/documentation/productlibrary/index.html?productID=62286"^]
    - Technischer Bericht zu NetApp ONTAP FlexGroup Volumeshttps://www.netapp.com/us/media/tr-4557.pdf["https://www.netapp.com/us/media/tr-4557.pdf"^]
- NetApp ONTAP AI
    - Design-Leitfaden: ONTAP AI mit DGX-1 und Cisco Networkinghttps://www.netapp.com/us/media/nva-1121-design.pdf["https://www.netapp.com/us/media/nva-1121-design.pdf"^]
    - ONTAP AI mit DGX-1 und Cisco Networking Deployment Guidehttps://www.netapp.com/us/media/nva-1121-deploy.pdf["https://www.netapp.com/us/media/nva-1121-deploy.pdf"^]
    - Design-Leitfaden: ONTAP AI mit DGX-1 und Mellanox Networkinghttp://www.netapp.com/us/media/nva-1138-design.pdf["http://www.netapp.com/us/media/nva-1138-design.pdf"^]
    - Design-Leitfaden: ONTAP AI mit DGX-2https://www.netapp.com/us/media/nva-1135-design.pdf["https://www.netapp.com/us/media/nva-1135-design.pdf"^]

Copyright-Informationen

Copyright © 2024 NetApp. Alle Rechte vorbehalten. Gedruckt in den USA. Dieses urheberrechtlich geschützte Dokument darf ohne die vorherige schriftliche Genehmigung des Urheberrechtsinhabers in keiner Form und durch keine Mittel – weder grafische noch elektronische oder mechanische, einschließlich Fotokopieren, Aufnehmen oder Speichern in einem elektronischen Abrufsystem – auch nicht in Teilen, vervielfältigt werden.

Software, die von urheberrechtlich geschütztem NetApp Material abgeleitet wird, unterliegt der folgenden Lizenz und dem folgenden Haftungsausschluss:

DIE VORLIEGENDE SOFTWARE WIRD IN DER VORLIEGENDEN FORM VON NETAPP ZUR VERFÜGUNG GESTELLT, D. H. OHNE JEGLICHE EXPLIZITE ODER IMPLIZITE GEWÄHRLEISTUNG, EINSCHLIESSLICH, JEDOCH NICHT BESCHRÄNKT AUF DIE STILLSCHWEIGENDE GEWÄHRLEISTUNG DER MARKTGÄNGIGKEIT UND EIGNUNG FÜR EINEN BESTIMMTEN ZWECK, DIE HIERMIT AUSGESCHLOSSEN WERDEN. NETAPP ÜBERNIMMT KEINERLEI HAFTUNG FÜR DIREKTE, INDIREKTE, ZUFÄLLIGE, BESONDERE, BEISPIELHAFTE SCHÄDEN ODER FOLGESCHÄDEN (EINSCHLIESSLICH, JEDOCH NICHT BESCHRÄNKT AUF DIE BESCHAFFUNG VON ERSATZWAREN ODER -DIENSTLEISTUNGEN, NUTZUNGS-, DATEN- ODER GEWINNVERLUSTE ODER UNTERBRECHUNG DES GESCHÄFTSBETRIEBS), UNABHÄNGIG DAVON, WIE SIE VERURSACHT WURDEN UND AUF WELCHER HAFTUNGSTHEORIE SIE BERUHEN, OB AUS VERTRAGLICH FESTGELEGTER HAFTUNG, VERSCHULDENSUNABHÄNGIGER HAFTUNG ODER DELIKTSHAFTUNG (EINSCHLIESSLICH FAHRLÄSSIGKEIT ODER AUF ANDEREM WEGE), DIE IN IRGENDEINER WEISE AUS DER NUTZUNG DIESER SOFTWARE RESULTIEREN, SELBST WENN AUF DIE MÖGLICHKEIT DERARTIGER SCHÄDEN HINGEWIESEN WURDE.

NetApp behält sich das Recht vor, die hierin beschriebenen Produkte jederzeit und ohne Vorankündigung zu ändern. NetApp übernimmt keine Verantwortung oder Haftung, die sich aus der Verwendung der hier beschriebenen Produkte ergibt, es sei denn, NetApp hat dem ausdrücklich in schriftlicher Form zugestimmt. Die Verwendung oder der Erwerb dieses Produkts stellt keine Lizenzierung im Rahmen eines Patentrechts, Markenrechts oder eines anderen Rechts an geistigem Eigentum von NetApp dar.

Das in diesem Dokument beschriebene Produkt kann durch ein oder mehrere US-amerikanische Patente, ausländische Patente oder anhängige Patentanmeldungen geschützt sein.

ERLÄUTERUNG ZU „RESTRICTED RIGHTS“: Nutzung, Vervielfältigung oder Offenlegung durch die US-Regierung unterliegt den Einschränkungen gemäß Unterabschnitt (b)(3) der Klausel „Rights in Technical Data – Noncommercial Items“ in DFARS 252.227-7013 (Februar 2014) und FAR 52.227-19 (Dezember 2007).

Die hierin enthaltenen Daten beziehen sich auf ein kommerzielles Produkt und/oder einen kommerziellen Service (wie in FAR 2.101 definiert) und sind Eigentum von NetApp, Inc. Alle technischen Daten und die Computersoftware von NetApp, die unter diesem Vertrag bereitgestellt werden, sind gewerblicher Natur und wurden ausschließlich unter Verwendung privater Mittel entwickelt. Die US-Regierung besitzt eine nicht ausschließliche, nicht übertragbare, nicht unterlizenzierbare, weltweite, limitierte unwiderrufliche Lizenz zur Nutzung der Daten nur in Verbindung mit und zur Unterstützung des Vertrags der US-Regierung, unter dem die Daten bereitgestellt wurden. Sofern in den vorliegenden Bedingungen nicht anders angegeben, dürfen die Daten ohne vorherige schriftliche Genehmigung von NetApp, Inc. nicht verwendet, offengelegt, vervielfältigt, geändert, aufgeführt oder angezeigt werden. Die Lizenzrechte der US-Regierung für das US-Verteidigungsministerium sind auf die in DFARS-Klausel 252.227-7015(b) (Februar 2014) genannten Rechte beschränkt.

Markeninformationen

NETAPP, das NETAPP Logo und die unter http://www.netapp.com/TM aufgeführten Marken sind Marken von NetApp, Inc. Andere Firmen und Produktnamen können Marken der jeweiligen Eigentümer sein.