



NVIDIA を使用した会話型 AI NetApp Solutions

NetApp
April 10, 2024

This PDF was generated from https://docs.netapp.com/ja-jp/netapp-solutions/ai/cainvidia_solution_overview.html on April 10, 2024. Always check docs.netapp.com for the latest.

目次

WP-7328 : 『 NetApp Conversational AI Using NVIDIA Jarvis 』	1
解決策の概要	1
解決策テクノロジー	3
概要	5
まとめ	19
謝辞	20
追加情報の検索場所	20

WP-7328 : 『 NetApp Conversational AI Using NVIDIA Jarvis 』

ネットアップ、Rick Huang、Sung-Han Lin、NVIDIA、Davide Onofrio

NVIDIA DGX システムファミリーは、エンタープライズ AI に特化した世界初の人工知能（AI）ベースシステムで構成されます。NetApp AFF ストレージシステムは、卓越したパフォーマンスと業界をリードするハイブリッドクラウドデータ管理機能を提供します。ネットアップと NVIDIA は提携を通じて、NetApp ONTAP AI リファレンスアーキテクチャを構築しました。このリファレンスアーキテクチャは、AI と機械学習（ML）のワークロード向けのターンキー解決策であり、エンタープライズクラスのパフォーマンス、信頼性、サポートを提供します。

このホワイトペーパーでは、さまざまな業種のさまざまなユースケースに対応した会話型 AI システムを構築するお客様にガイダンスを提供します。これには、NVIDIA Jarvis を使用したシステムの導入に関する情報が含まれています。テストは NVIDIA DGX ステーションと NetApp AFF A220 ストレージシステムを使用して実施しました。

解決策の対象となるグループは次のとおりです。

- AI 開発のためのソリューションを設計するエンタープライズアーキテクトなどの会話型 AI のユースケースに適したモデルとソフトウェア バーチャル・リテール・アシスタント
- データサイエンティストは、言語モデリングを効率的に実現する方法を探しています 能力開発の目標
- テキストデータの保守と処理を担当するデータエンジニア お客様からの質問や会話の記録など
- エグゼクティブや IT の意思決定者、および関心のあるビジネスリーダー 会話型 AI のエクスペリエンスを変革し、最速の時間を実現できます AI への取り組みから市場に投入

解決策の概要

NetApp ONTAP AIとBlueXPのコピーと同期

ネットアップと NVIDIA は、NVIDIA DGX システムとネットアップのクラウド対応ストレージシステムを基盤とする NetApp ONTAP AI アーキテクチャを開発、検証しました。このリファレンスアーキテクチャには、IT 組織に次のようなメリットがあります。

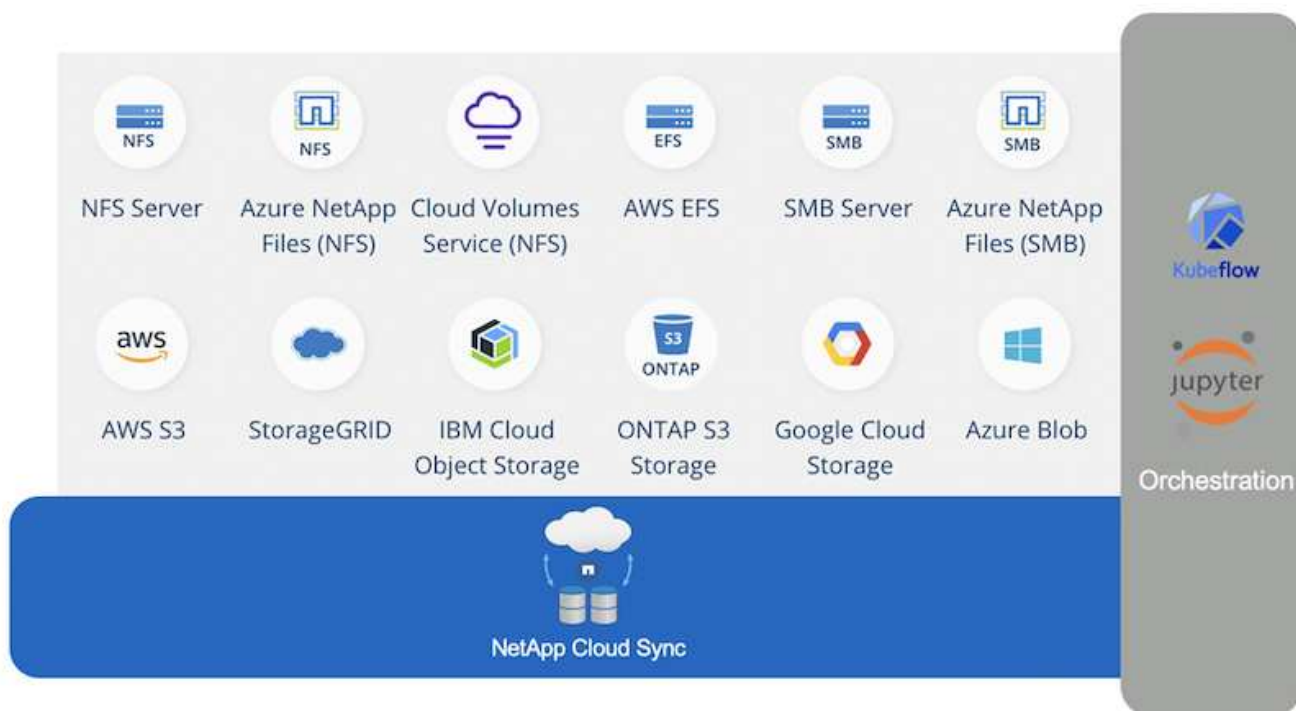
- 設計の複雑さを解消
- コンピューティングとストレージを個別に拡張できます
- 小規模構成から始めて、シームレスに拡張できます
- さまざまなパフォーマンスとコストの観点から、幅広いストレージオプションを提供 NetApp ONTAP AI は、DGX システムと NetApp AFF A220 ストレージシステムを最先端のネットワーク機能と緊密に統合します。NetApp ONTAP AI システムと DGX システムでは、設計の複雑さと推測に頼らず、AI 導入を簡易化できます。お客様は小規模構成から始めて、システムを中断なく拡張できます。同時に、エッジ、コア、クラウドにわたってデータをインテリジェントに管理できます。

NetApp BlueXPのコピーと同期機能を使用すると、2つのNFS共有、2つのCIFS共有、1つのファイル共有

とAmazon S3、Amazon Elastic File System（EFS）、Azure Blob Storageの間など、さまざまなプロトコルを使用してデータを簡単に移動できます。アクティブ / アクティブ処理とは、ソースとターゲットの両方と同時に作業を継続し、必要に応じてデータの変更を段階的に同期することを意味します。BlueXPのCopy and Syncは、オンプレミスでもクラウドベースでも、あらゆるソースシステムとデスティネーションシステムの間でデータを移動して差分同期できるため、データの利用方法が多様化します。オンプレミスのシステム間でのデータ移行、クラウドへのオンボーディングやクラウドへの移行、コラボレーションとデータ分析などのすべての作業を容易に実現できます。次の図は、使用可能なソースとデスティネーションを示しています。

会話型AIシステムでは、開発者はBlueXPのコピーと同期を活用して会話履歴をクラウドからデータセンターにアーカイブし、自然言語処理（NLP）モデルのオフライントレーニングを実現できます。より多くのインテントを認識するためのトレーニングモデルによって、会話型AIシステムは、エンドユーザーからのより複雑な質問にも対応できるようになります。

NVIDIA Jarvis マルチモーダルフレームワーク



"NVIDIA Jarvis" 会話型 AI サービスを構築するためのエンドツーエンドのフレームワークです。GPU 向けに最適化された次のサービスが含まれています。

- 自動音声認識（ASR）
- 自然言語理解（NLU）
- ドメイン固有のフルフィルメントサービスとの統合
- テキスト / スピーチ（TTS）
- コンピュータビジョン（CV）ジャービスベースのサービスは、最先端のディープラーニングモデルを使用して、リアルタイムの会話型 AI の複雑で困難なタスクに対処します。エンドユーザーとのリアルタイムかつ自然な対話を可能にするには、モデルが 300 ミリ秒未満で計算を完了する必要があります。自然な相互作用は困難であり、マルチモーダル感覚を統合する必要があります。モデルパイプラインも複雑で、上記のサービス全体で調整が必要です。

Jarvis は、エンドツーエンドのディープラーニングパイプラインを使用する、マルチモーダル会話型 AI サービスを構築するための、完全に高速化されたアプリケーションフレームワークです。Jarvis フレームワークに

は、音声、ビジョン、および NLU タスク向けに、事前にトレーニングされた会話型 AI モデル、ツール、最適化されたエンドツーエンドサービスが含まれます。AI サービスに加えて、Jarvis ではビジョン、オーディオ、およびその他のセンサー入力を同時に融合し、仮想アシスタント、マルチユーザーのディアゼーション、コールセンターアシスタントなどのアプリケーションでマルチコンテキスト会話などの機能を提供できます。

NVIDIA Nemo

"NVIDIA Nemo" は、使いやすいアプリケーションプログラミングインターフェイス（API）を使用して、GPU によって高速化された最先端の会話型 AI モデルを構築、トレーニング、微調整するためのオープンソース Python ツールキットです。Nemo は、NVIDIA GPU で Tensor コアを使用して精度の高いコンピューティングを実行し、複数の GPU に簡単にスケールアップして、トレーニングのパフォーマンスを最大限に高めることができます。Nemo は、医療、金融、小売、通信など、さまざまな業界のさまざまな業界で、ビデオ通話の文字変換、インテリジェントビデオアシスタント、自動コールセンターサポートなどのリアルタイム ASR、NLP、TTS アプリケーションのモデルを構築するために使用されます。

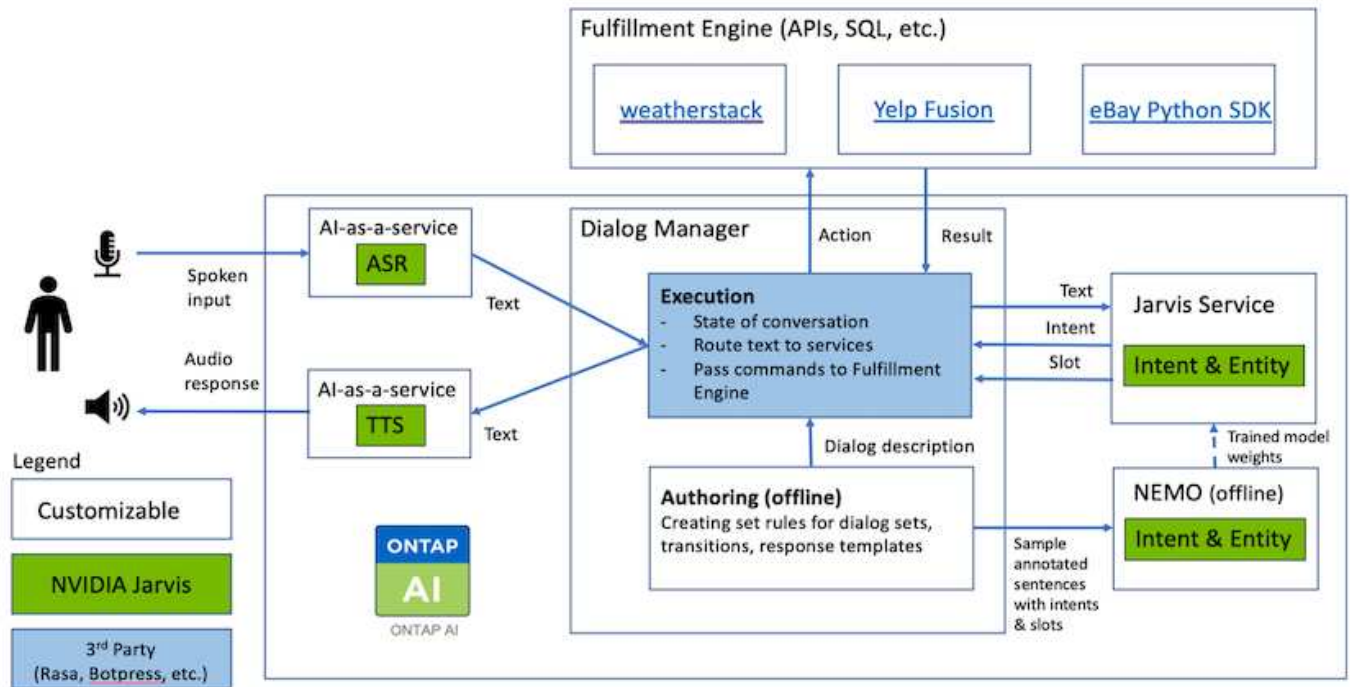
Nemo を使用して、アーカイブされた会話履歴のユーザ質問から複雑な_intentを認識するモデルをトレーニングしました。このトレーニングは、Jarvis が提供したもの以外にも、小売バーチャルアシスタントの機能を拡張します。

小売業のユースケースの概要

NVIDIA Jarvis を使用して、スピーチやテキスト入力を受け付け、天気、関心のあるポイント、在庫価格に関する質問に回答できる仮想小売アシスタントを構築しました。会話型 AI システムでは、たとえば、天気や関心のある場所を指定していない場合は、フォローアップの質問をして会話の流れを記憶することができます。また、「タイ料理」や「ノートパソコンのメモリ」などの複雑なエンティティも認識します。「ロサンゼルスで来週雨が降るだろうか？」など、自然言語の質問を理解しています。小売バーチャルアシスタントのデモンストレーションは、にあります ["小売ユースケースの状態とフローをカスタマイズします"](#)。

解決策テクノロジー

次の図は、提案された会話型 AI システムアーキテクチャを示しています。システムは、音声信号またはテキスト入力で操作できます。音声入力が発出された場合、Jarvis As-a-Service（AlaaS）は ASR を実行して、Dialog Manager 用のテキストを生成します。Dialog Manager は、会話の状態を記憶し、テキストを対応するサービスにルーティングし、Fulfillment Engine にコマンドを渡します。Jarvis NLP サービスは、テキストを受け取り、intentとエンティティを認識し、それらのintentとエンティティスロットをダイアログマネージャーに出力します。ダイアログマネージャーは、アクションをフルフィルメントエンジンに送信します。フルフィルメントエンジンは、回答ユーザーが照会するサードパーティ API または SQL データベースで構成されています。フルフィルメントエンジンから結果を受け取った後、Dialog Manager はテキストを Jarvis TTS AlaaS にルーティングして、エンドユーザーの音声応答を生成します。NLP サービスがシステムとの対話をより多くのユーザーが行うように改善されるように、会話履歴をアーカイブし、intentや Nemo トレーニング用のスロットに注釈を付けることができます。



ハードウェア要件

この解決策は、1 台の DGX ステーションと 1 台の AFF A220 ストレージシステムで検証済みです。Jarvis では、ディープニューラルネットワークの計算に T4 GPU または V100 GPU のいずれかが必要です。

次の表に、テストで解決策を実装するために必要なハードウェアコンポーネントを示します。

ハードウェア	数量
T4 または V100 GPU	1.
NVIDIA DGX ステーション	1.

ソフトウェア要件

次の表に、テストで解決策を実装するために必要なソフトウェアコンポーネントを示します。

ソフトウェア	バージョンまたはその他の情報
NetApp ONTAP データ管理ソフトウェア	9.6
Cisco NX-OS スイッチのファームウェア	7.0 (3) I6 (1)
NVIDIA DGX OS	4.0.4 - Ubuntu 18.04 LTS
NVIDIA Jarvis フレームワーク	EA v0.2 の 1 つです
NVIDIA Nemo	nvcr.io/nvidia / Nemo : v0.10
Docker コンテナプラットフォーム	18.06.1-CE [e68fc7a]

概要

このセクションでは、仮想小売アシスタントの実装について詳しく説明します。

Jarvis の展開

にサインアップできます ["Jarvis Early Access プログラム"](#) NVIDIA GPU Cloud (NGC) のジャービスコンテナにアクセスするため。NVIDIA から資格情報を受け取った後、以下の手順を使用して Jarvis を展開できます。

1. NGC へのサインオン
2. NGC に組織を設定します : 'ea -2- ジャービス'.
3. Jarvis EA v0.2 資産の検索 : Jarvis コンテナは「プライベートレジストリ」>「組織コンテナ」にあります。
4. 「Jarvis」を選択します。「Model Scripts」に移動し、「Jarvis Quick Start」をクリックします
5. すべてのアセットが正しく動作していることを確認します。
6. 独自のアプリケーションを構築するためのドキュメントを検索します。PDF は「Model Scripts」>「Jarvis Documentation」>「ファイルブラウザ」にあります。

小売ユースケースの状態とフローをカスタマイズします

特定のユースケースに合わせてダイアログマネージャーの状態とフローをカスタマイズできます。小売業の例では、次の 4 つの YAML ファイルを使用して、異なるインテントに従って会話を誘導します。

各ファイルについて、次のファイル名と概要のリストを用意します。

- `main_flow.yml`: 主な会話の流れと状態を定義し、必要に応じて他の 3 つの YAML ファイルにフローを指示します。
- `retail_flow.yml`: 小売業または関心のある点に関する質問に関連する状態を含みます。システムは最も近い店の情報、または与えられた項目の価格を提供する。
- `weater_flow.yml`: 天気に関する質問に関連する州を含みます。場所を特定できない場合は、フォローアップの質問をして明確にします。
- `error_flow.yml`: ユーザーのインテントが上記の 3 つの YAML ファイルに入っていないケースを処理します。エラーメッセージを表示した後、システムはユーザーの質問を受け入れるように再経路化します。次のセクションでは、これらの YAML ファイルの詳細な定義を示します。

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

```

```

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

```

フルフィルメントエンジンとしてサードパーティ **API** に接続します

次のサードパーティ API を欠品補充エンジンとして回答の質問に関連付けました。

- ["WeatherStack API"](#): 指定された場所の天候、温度、降雨および雪を返す。
- ["Yelp Fusion API"](#): 指定された場所で最も近いストア情報を返します。
- ["eBay Python SDK"](#): 指定されたアイテムの価格を返します。

NetApp Retail Assistant のデモ

NetApp Retail Assistant （奈良）のデモビデオを録画しました。

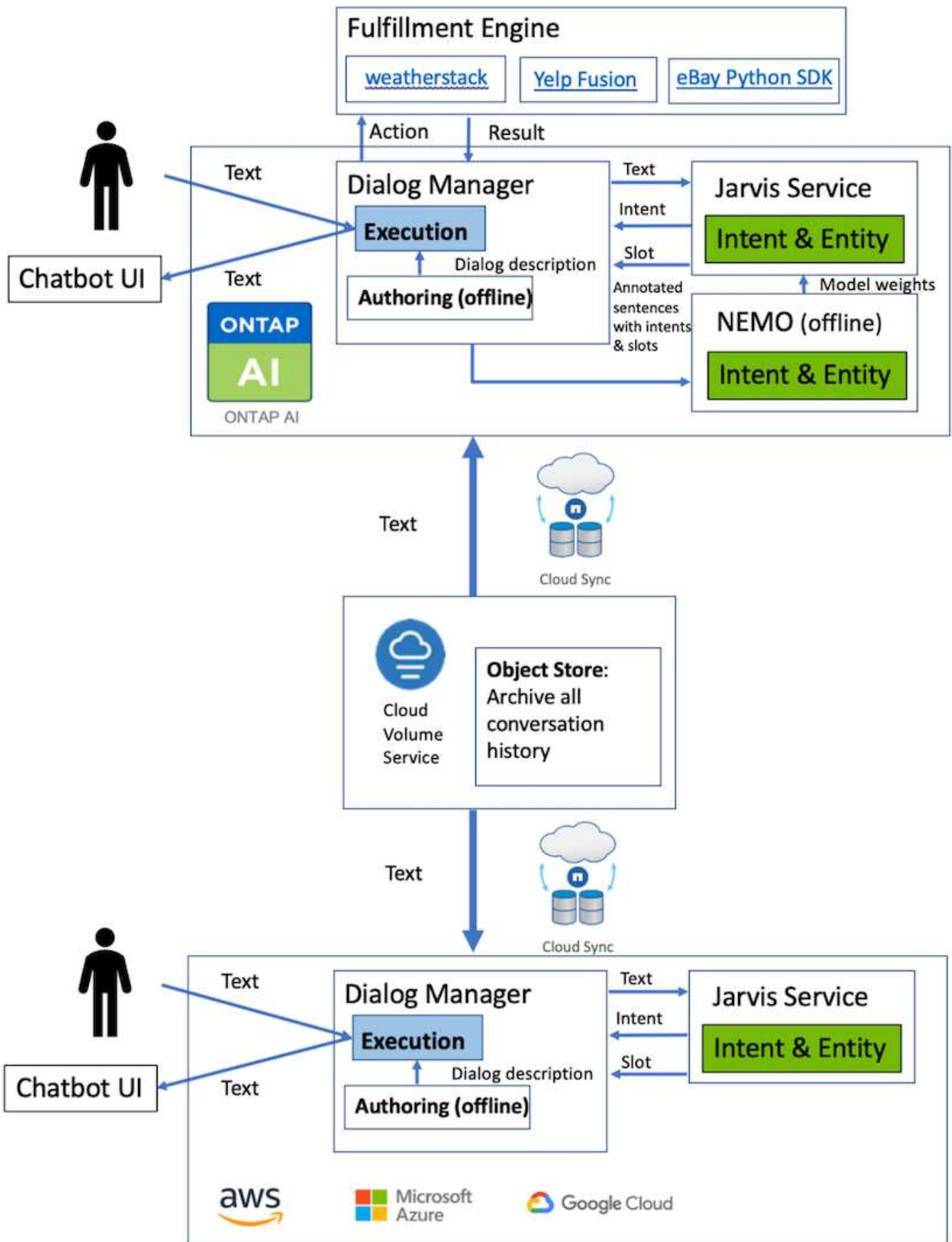
奈良のデモ映像

[奈良のデモ映像](#)



NetApp BlueXPのコピーと同期を使用して会話履歴をアーカイブ

会話履歴を1日1回CSVファイルにダンプすることで、BlueXPのコピーと同期を活用してローカルストレージにログファイルをダウンロードできます。次の図は、Jarvisをオンプレミスとパブリッククラウドに導入し、BlueXPのCopy and Syncを使用してNemoトレーニングに関する会話履歴を送信するアーキテクチャを示しています。Nemo の訓練の詳細はセクションで見つけることができる ["Nemo トレーニングを使用してインテントモデルを拡張する"](#)。



Nemo トレーニングを使用して_intentモデルを拡張する

NVIDIA Nemo は、会話型 AI アプリケーションを作成するために NVIDIA が開発したツールキットです。このツールキットには、ASR、NLP、TTS 用のトレーニング済みモジュールのコレクションが含まれています。これにより、研究者やデータサイエンティストは複雑なニューラルネットワークアーキテクチャを簡単に構築し、独自のアプリケーションの設計に集中できるようになります。

前の例で示したように、奈良は限られたタイプの質問しか処理できない。これは、トレーニング済みの NLP モデルでは、これらのタイプの質問についてのみトレーニングを受けているためです。奈良がより幅広い質問に対応できるようにするには、独自のデータセットを使って再トレーニングする必要があります。ここでは、Nemo を使用して NLP モデルを拡張し、要件を満たす方法を示します。まず、奈良から収集したログを Nemo の形式に変換し、NLP モデルを強化するためのデータセットを使って学習します。

モデル

私たちの目標は、ユーザーの好みに基づいてアイテムを並べ替えることです。たとえば、奈良に最高級の寿司レストランを提案したり、奈良に最も安い価格のジーンズを探してもらうようにしたりすることができます。このためには、Nemo で提供されている_intent検出とスロット充填モデルをトレーニングモデルとして使用します。このモデルにより、奈良は検索の好みを理解することができます。

データの準備

モデルをトレーニングするには、このタイプの質問のデータセットを収集し、Nemo 形式に変換します。ここでは、モデルのトレーニングに使用するファイルをリストしました。

dict.intents.csv

このファイルには、Nemo に理解してもらいたいすべての_intentが一覧表示されます。ここでは、主な_intentが 2 つあり、1 つの_intentは、主な_intentのどれにも当てはまらない質問を分類するために使用されます。

```
price_check
find_the_store
unknown
```

dict.slots.csv

このファイルには、トレーニングの質問にラベルを付けることができるすべてのスロットが記載されています。

```
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

これが主なトレーニングデータセットです。各行は、dict.intent.csv ファイルの_intentカテゴリーのリストに続く質問から始まります。ラベルはゼロから列挙されます。

train slots.tsv

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

モデルのトレーニング

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

次に、次のコマンドを使用してコンテナを起動します。このコマンドでは、軽量なトレーニング用の演習であるため、コンテナで使用する GPU は 1 つに制限されます（GPU ID = 1）。また、ローカルワークスペース / ワークスペース / Nemo/ をコンテナ / Nemo 内のフォルダにマッピングします。

```
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
```

コンテナ内では、事前にトレーニングされたオリジナルの BERT モデルから開始する場合、次のコマンドを使用してトレーニング手順を開始できます。data_dir は、トレーニングデータのパスを設定する引数です。work_dir では 'チェックポイント・ファイルを保存する場所を設定できます

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

新しいトレーニングデータセットがあり、以前のモデルを改善したい場合は、次のコマンドを使用して停止した時点から続行できます。checkpoint_dir は '前のチェックポイント・フォルダへのパスを取得します

```
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
```

モデルを推論します

トレーニング済みモデルのパフォーマンスは、一定の期間の経過後に検証する必要があります。次のコマンドを使用すると、1 つずつクエリをテストできます。たとえば、このコマンドでは、モデルがクエリの意図を正しく識別できるかどうかを確認します。クエリの目的は、「ここで最高のパスタを取得できる」です。

```
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
```

次に、推論からの出力を示します。出力では、トレーニング済みモデルが意図を正しく予測し、関心のあるキーワードを返すことができます。これらのキーワードを使うことで、奈良はユーザが欲しいものを検索し、より正確な検索を行うことができるようになります。

```
[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
```

まとめ

真の会話型 AI システムは、人間のような対話を行い、文脈を理解し、インテリジェントな応答を提供します。このような AI モデルは、多くの場合、巨大で非常に複雑です。NVIDIA GPU とネットアップストレージを使用することで、最新の大規模な言語モデルをトレーニングし、最適化して推論を迅速に実行できます。これは、大規模で複雑な AI モデルと比べて高速な AI モデルのトレードオフを終えるための大きなストライドです。GPU に最適化された言語理解モデルは、医療、小売、金融サービスなどの業界向け AI アプリケーションに統合でき、高度なデジタル音声アシスタントをスマートスピーカーやカスタマーサービスラインに搭載できます。これらの高品質な会話型 AI システムにより、さまざまな業種の企業が、お客様との取引においてこれまで達成できなかったパーソナライズされたサービスを提供できます。

Jarvis は、バーチャルアシスタント、デジタルアバター、マルチモーダルセンサー Fusion（ASR/NLP/TTS とフュージョンされた CV）、または ASR/NLP/TTS/CV スタンドアロンの使用例（転写など）の導入を可能にします。私たちは、天気、関心のあるポイント、在庫価格に関する回答の質問を行えるバーチャル小売アシスタントを開発しました。また、BlueXP CopyとSyncを使用して会話履歴をアーカイブし、新しいデータについてNemoモデルをトレーニングすることで、会話型AIシステムの自然言語理解機能を向上させる方法についてもデモンストレーションを行いました。

謝辞

このホワイトペーパーに貢献したことを、NVIDIA の著名な同僚によって認められたことに感謝の意を表します。Davide Onfrio、Alex Qi、Sicong Ji、Marty Jain、Robert Sohigian。また、ネットアップの主要チームメンバーである Santosh Rao、David Arnette、Michael Oglesby、Brent Davis、Andy Sayare の協力も感謝しています。Erik Mulder 氏、Mike McNamara 氏。

本ホワイトペーパーの作成に大きく貢献した洞察と専門知識を提供したすべての方々に心から感謝します。

追加情報の検索場所

このドキュメントに記載されている情報の詳細については、次のリソースを参照してください。

- NVIDIA DGX Station、V100 GPU、GPU Cloud
 - NVIDIA DGX ステーション<https://www.nvidia.com/en-us/data-center/dgx-station/>[\["https://www.nvidia.com/en-us/data-center/dgx-station/"^\]](https://www.nvidia.com/en-us/data-center/dgx-station/)
 - NVIDIA V100 Tensor コア GPU<https://www.nvidia.com/en-us/data-center/tesla-v100/>[\["https://www.nvidia.com/en-us/data-center/tesla-v100/"^\]](https://www.nvidia.com/en-us/data-center/tesla-v100/)
 - NVIDIA NGC<https://www.nvidia.com/en-us/gpu-cloud/>[\["https://www.nvidia.com/en-us/gpu-cloud/"^\]](https://www.nvidia.com/en-us/gpu-cloud/)
- NVIDIA Jarvis マルチモーダルフレームワーク
 - NVIDIA Jarvis<https://developer.nvidia.com/nvidia-jarvis>[\["https://developer.nvidia.com/nvidia-jarvis"^\]](https://developer.nvidia.com/nvidia-jarvis)
 - NVIDIA Jarvis Early Access<https://developer.nvidia.com/nvidia-jarvis-early-access>[\["https://developer.nvidia.com/nvidia-jarvis-early-access"^\]](https://developer.nvidia.com/nvidia-jarvis-early-access)
- NVIDIA Nemo
 - NVIDIA Nemo<https://developer.nvidia.com/nvidia-nemo>[\["https://developer.nvidia.com/nvidia-nemo"^\]](https://developer.nvidia.com/nvidia-nemo)
 - 開発者ガイド<https://nvidia.github.io/NeMo/>[\["https://nvidia.github.io/NeMo/"^\]](https://nvidia.github.io/NeMo/)
- NetApp AFF システム
 - NetApp AFF A シリーズのデータシート<https://www.netapp.com/us/media/ds-3582.pdf>[\["https://www.netapp.com/us/media/ds-3582.pdf"^\]](https://www.netapp.com/us/media/ds-3582.pdf)
 - ネットアップの All Flash FAS 向けフラッシュソリューションの利点<https://www.netapp.com/us/media/ds-3733.pdf>[\["https://www.netapp.com/us/media/ds-3733.pdf"^\]](https://www.netapp.com/us/media/ds-3733.pdf)
 - ONTAP 9 情報ライブラリ<http://mysupport.netapp.com/documentation/productlibrary/index.html?productID=62286>[\["http://mysupport.netapp.com/documentation/productlibrary/index.html?productID=62286"^\]](http://mysupport.netapp.com/documentation/productlibrary/index.html?productID=62286)

- NetApp ONTAP FlexGroup Volume テクニカルレポート <https://www.netapp.com/us/media/tr-4557.pdf>["<https://www.netapp.com/us/media/tr-4557.pdf>"]
- NetApp ONTAP AI
 - DGX-1 と Cisco Networking Design Guide による ONTAP AI <https://www.netapp.com/us/media/nva-1121-design.pdf>["<https://www.netapp.com/us/media/nva-1121-design.pdf>"]
 - DGX-1 と Cisco Networking Deployment Guide を使用した ONTAP AI <https://www.netapp.com/us/media/nva-1121-deploy.pdf>["<https://www.netapp.com/us/media/nva-1121-deploy.pdf>"]
 - DGX-1 と Mellanox のネットワーキング設計ガイドで構成される ONTAP AI <http://www.netapp.com/us/media/nva-1138-design.pdf>["<http://www.netapp.com/us/media/nva-1138-design.pdf>"]
 - DGX-2 を使用した ONTAP AI 設計ガイド <https://www.netapp.com/us/media/nva-1135-design.pdf>["<https://www.netapp.com/us/media/nva-1135-design.pdf>"]

著作権に関する情報

Copyright © 2024 NetApp, Inc. All Rights Reserved. Printed in the U.S. このドキュメントは著作権によって保護されています。著作権所有者の書面による事前承諾がある場合を除き、画像媒体、電子媒体、および写真複写、記録媒体、テープ媒体、電子検索システムへの組み込みを含む機械媒体など、いかなる形式および方法による複製も禁止します。

ネットアップの著作物から派生したソフトウェアは、次に示す使用許諾条項および免責条項の対象となります。

このソフトウェアは、ネットアップによって「現状のまま」提供されています。ネットアップは明示的な保証、または商品性および特定目的に対する適合性の暗示的保証を含み、かつこれに限定されないいかなる暗示的な保証も行いません。ネットアップは、代替品または代替サービスの調達、使用不能、データ損失、利益損失、業務中断を含み、かつこれに限定されない、このソフトウェアの使用により生じたすべての直接的損害、間接的損害、偶発的損害、特別損害、懲罰的損害、必然的損害の発生に対して、損失の発生の可能性が通知されていたとしても、その発生理由、根拠とする責任論、契約の有無、厳格責任、不法行為（過失またはそうでない場合を含む）にかかわらず、一切の責任を負いません。

ネットアップは、ここに記載されているすべての製品に対する変更を随時、予告なく行う権利を保有します。ネットアップによる明示的な書面による合意がある場合を除き、ここに記載されている製品の使用により生じる責任および義務に対して、ネットアップは責任を負いません。この製品の使用または購入は、ネットアップの特許権、商標権、または他の知的所有権に基づくライセンスの供与とはみなされません。

このマニュアルに記載されている製品は、1つ以上の米国特許、その他の国の特許、および出願中の特許によって保護されている場合があります。

権利の制限について：政府による使用、複製、開示は、DFARS 252.227-7013（2014年2月）およびFAR 5252.227-19（2007年12月）のRights in Technical Data -Noncommercial Items（技術データ - 非商用品目に関する諸権利）条項の(b)(3)項、に規定された制限が適用されます。

本書に含まれるデータは商用製品および / または商用サービス（FAR 2.101の定義に基づく）に関係し、データの所有権はNetApp, Inc.にあります。本契約に基づき提供されるすべてのネットアップの技術データおよびコンピュータ ソフトウェアは、商用目的であり、私費のみで開発されたものです。米国政府は本データに対し、非独占的かつ移転およびサブライセンス不可で、全世界を対象とする取り消し不能の制限付き使用权を有し、本データの提供の根拠となった米国政府契約に関連し、当該契約の裏付けとする場合にのみ本データを使用できます。前述の場合を除き、NetApp, Inc.の書面による許可を事前に得ることなく、本データを使用、開示、転載、改変するほか、上演または展示することはできません。国防総省にかかる米国政府のデータ使用权については、DFARS 252.227-7015(b)項（2014年2月）で定められた権利のみが認められます。

商標に関する情報

NetApp、NetAppのロゴ、<http://www.netapp.com/TM>に記載されているマークは、NetApp, Inc.の商標です。その他の会社名と製品名は、それを所有する各社の商標である場合があります。