Jarvis、BlueXP Copy and Sync、Nemoを使用して仮想アシスタントを構築

NetApp Solutions

NetApp
September 10, 2024

目次

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「 M odel 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
```

weater_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
```

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

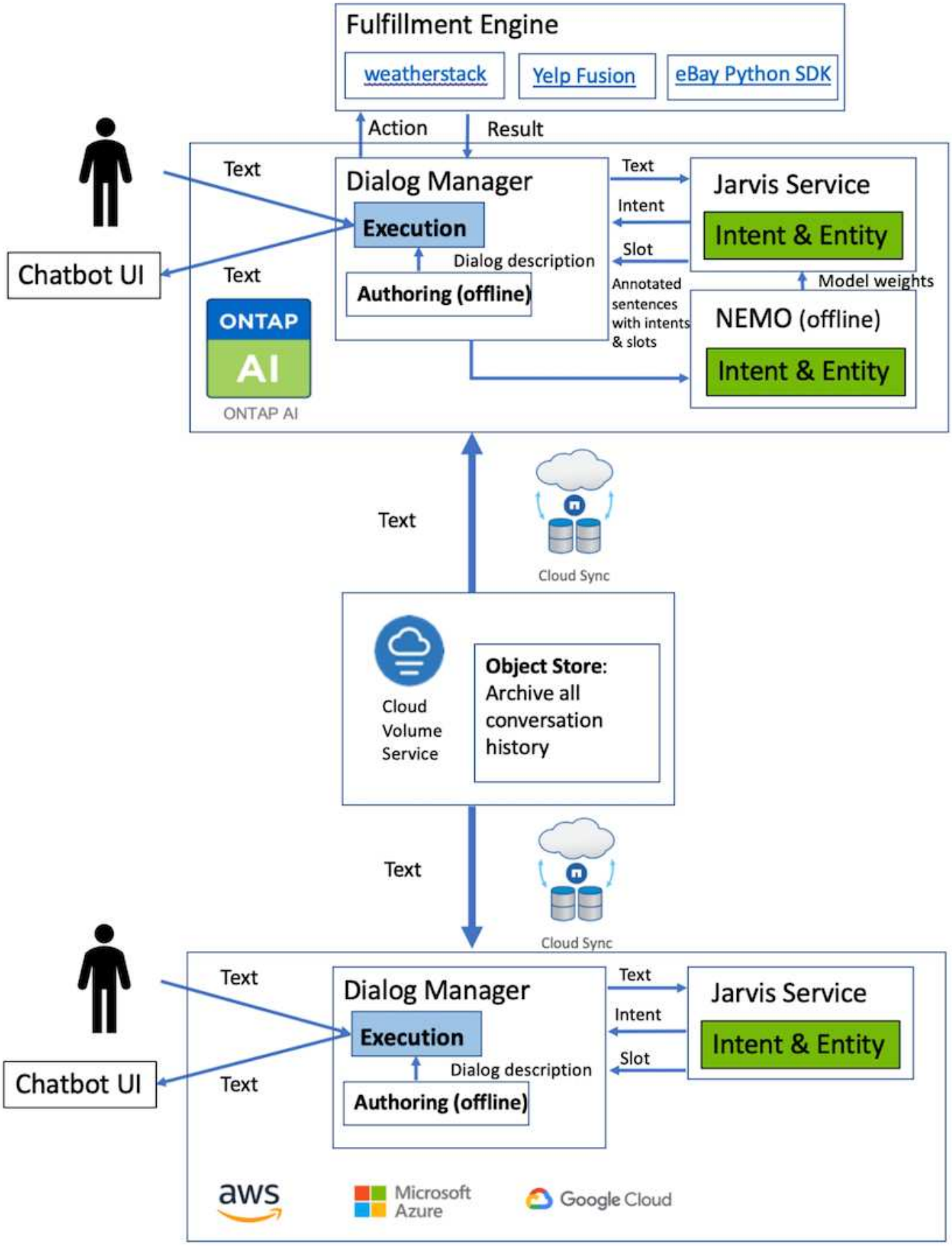
NetApp Retail Assistant （奈良）のデモビデオを録画しました。

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"。


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

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

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 で提供されているインテント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る。

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インテントが一覧表示されます。ここでは、主なインテントが 2 つあり、 1 つのインテントは、主なインテント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
```

鉄道 **.tsv**

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 ファイルのインテントカテゴリ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
```