

Developing Applications - Bots

Module 2 Developing a smarter Bot Student Lab Manual

Lab 3: Bot Framework and Luis

Introduction

Orange Networks GmbH

Sachsenteor 26
21029 Hamburg

Lyonel-Feininger-Strasse 28
81677 München

info@orange-networks.de
<http://www.orange-networks.de>
Telefon: +49 (0) 40 739 237 20
Telefax: +49 (0) 40 739 237 02

Geschäftsführer:
Oliver Mohr
Andreas Riedl
Matthias Seitle

In this lab, you will go through the creation and the integration of LUIS (Language Understanding Intelligent Service) Model with the bot application to make it smarter:

- Create a LUIS application
- Build and train a LUIS application
- Test and publish the LUIS application
- Integrate LUIS with Bot

Objectives

After completing this lab, you will be able to:

- Create a LUIS app
- Perform the usual operations for LUIS apps
- Integrate LUIS with Bot

Prerequisites

- Visual Studio 2017 Community
- Visual Studio Code (for Node.js)
- Latest Node.js with Node Package Manager. Download it from [here](#).
- Bot Framework Emulator

Estimated Time to Complete This Lab

60 minutes

For More Information

Get started with LUIS: <https://docs.microsoft.com/en-us/azure/cognitive-services/luis/luis-get-started-create-app>

Train and test your LUIS app: <https://docs.microsoft.com/en-us/azure/cognitive-services/luis/train-test>

Bot builder and Recognizing intents and entities with LUIS:

<https://docs.microsoft.com/en-us/bot-framework/dotnet/bot-builder-dotnet-luis-dialogs>

Exercise 1: Create a LUIS Application

Introduction

This exercise will guide you to create your first Language Understanding Intelligent Service (LUIS) app. When you're finished, you'll have a LUIS endpoint up and running in the cloud.

Objectives

After completing this lab, you will be able to:

- Create or import a LUIS app
- Add intents
- Add utterances
- Add entities
- Improve performance using features
- Train and test your model
- Publish the application endpoint

Prerequisites

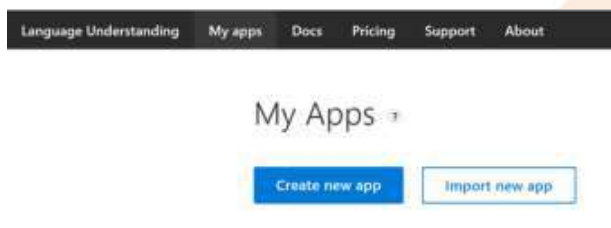
None

Scenario

You are going to create a search hotel app, that helps you search hotel and get the hotels reviews. If you would like to import an app, Go to the home page, <http://www.luis.ai>, and log in , click on **import app** and select the pre-build file **LuisBot.json** from the assets folder of the current lab. After that you can move directly to the second exercise.

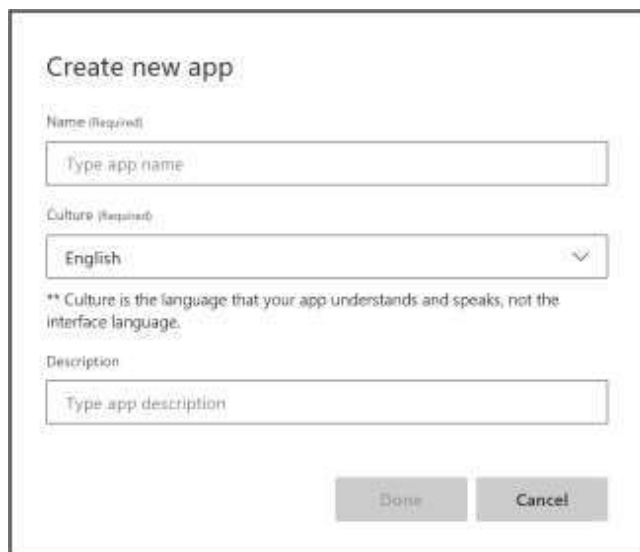
Task 1: Create a new app

The first step to use LUIS is to create or import an application. Go to the home page, <http://www.luis.ai>, and log in. (if you don't have a LUIS account you can create one) After you logged in to your LUIS account you'll be able to create or Import an Existing Application



You can create and manage your applications on **My Apps** page. You can always access this page by clicking **My Apps** on the top navigation bar of the [LUIS web page](#).

1. On the **My Apps** page, click **Create new App**.
2. In the dialog box, name your application "LuisBot".



Create new app

Name (required)
Type app name

Culture (required)
English

** Culture is the language that your app understands and speaks, not the interface language.

Description
Type app description

Done Cancel

3. Choose your application culture (for LuisBot app, we'll choose English), and then click **Done**.

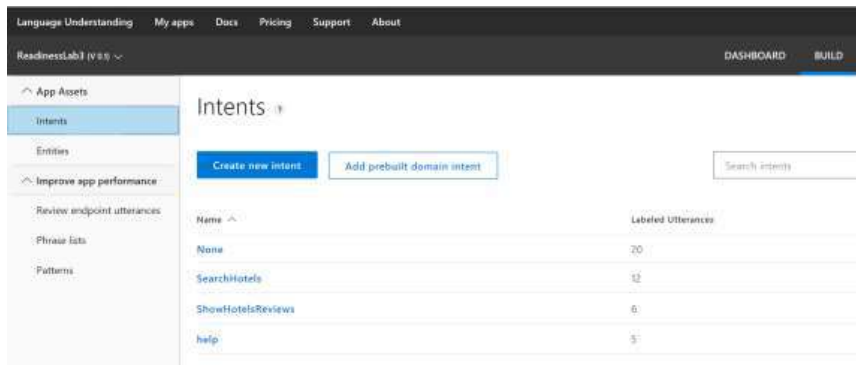
NB: The culture cannot be changed once the application is created.

LUIS creates the **LuisBot** app and opens its main page with navigation links in the left panel to move through your app pages to define data and work on your app.

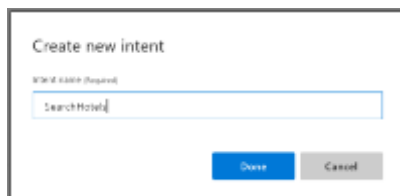
Task 2: Add intents

Your first task in the app is to add intents. Intent is a purpose or goal expressed in a user's input, or utterances. Intents match user requests with the actions that should be taken by your app, so you must add intents to help your app understand user requests and respond to them.

1. Open your app (for example, LuisBot) by clicking its name on **My Apps** page, and then click **Intents** in the left panel.



2. On the **Intents** page, click **Create new intent**.

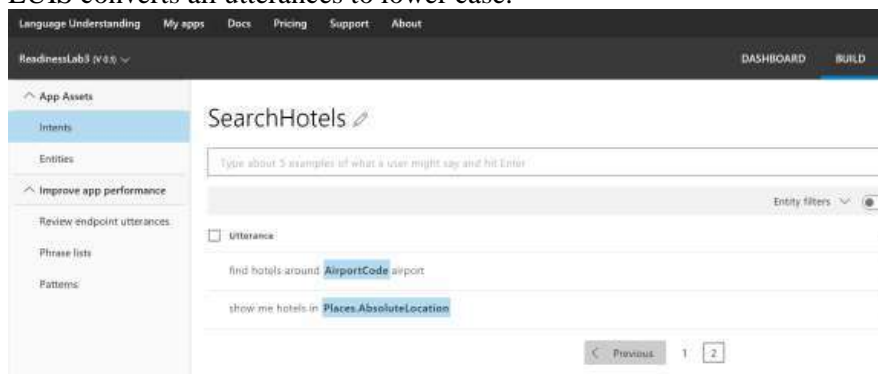


3. In the **Add Intent** dialog box, type the intent name "SearchHotels" and click **Save**.
4. This takes you directly to the intent details page of the newly added intent "**SearchHotels**", to add utterances for this intent.
5. Repeat the previous step to add two more intents: **Help** and **ShowHotelsReviews**

Task 3: Add utterance

To train LUIS, you need to add example utterances for each intent in your app. LUIS learns from these utterances and your app will be able to generalize and understand similar contexts. By constantly adding more utterances and labeling them, you are enhancing your application's language learning experience.

1. To add an **utterance** for the **SearchHotels** intent, type for example “**can you show me hotels from Paris?**” as a new utterance in the text box, and then press Enter. Note that LUIS converts all utterances to lower case.



2. Repeat the previous step to add more example utterances, examples:
 - look for hotels in Miami
 - search for hotels in seattle
 - show me hotels in California
 - best hotels in seattle
 - hotels in los angeles
 - can you show me hotels from los angeles?
 - find hotels near eze
 - where can i stay near nnn?
 - show hotels near Orly airport
 - find hotels around CDG airport
 - ...
3. Click Save to save the added utterances in the utterances list.
4. Repeat the steps of the task 3 to add utterances for **Help** intent with adding
 - What are the available options?
 - What can i do ?
 - Help me
 - I need help
 - ..

5. Repeat the steps of the task 3 to add utterances for **ShowHotelsReviews** intent with adding
 - What are the reviews of Sofitel Rabat ?
 - Can you show me the reviews of Sheraton Club des pins
 - can i see the reviews of extended bot hotel?
 - Find reviews of hotel xya
 - Show me reviews of the amazing hotel
 - Can you show me the reviews of the amazing resort & hotel
- ...

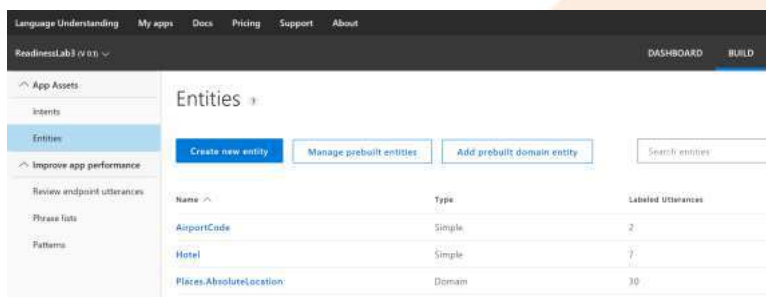
Task 4: Add Entities

Entities are key data in your application's domain. An entity represents a class including a collection of similar objects (places, things, people, events, or concepts). Entities describe information relevant to the intent, and sometimes they are essential for your app to perform its task.

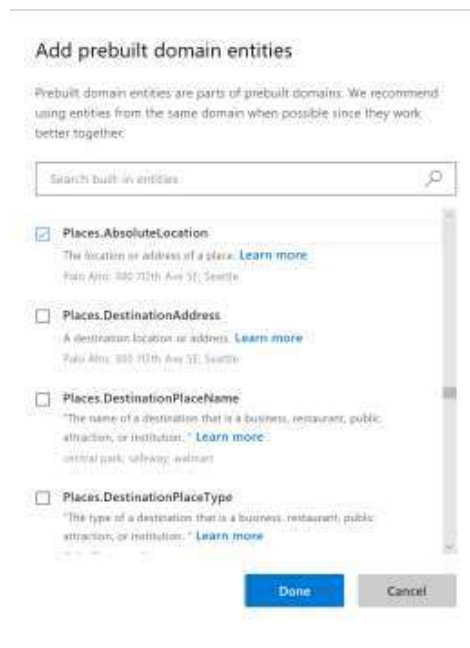
LUIS provides a set of prebuilt, system-defined entities. These entities cover many common concepts such as date, age, temperature, percentage, dimension, cardinal and ordinal numbers.

To add a prebuilt domain entity:

1. Open the **LuisBot** app by clicking its name on **My Apps** page, and then click **Entities** in the left panel.
2. On the **Entities** page, click **Add prebuilt domain entity**.

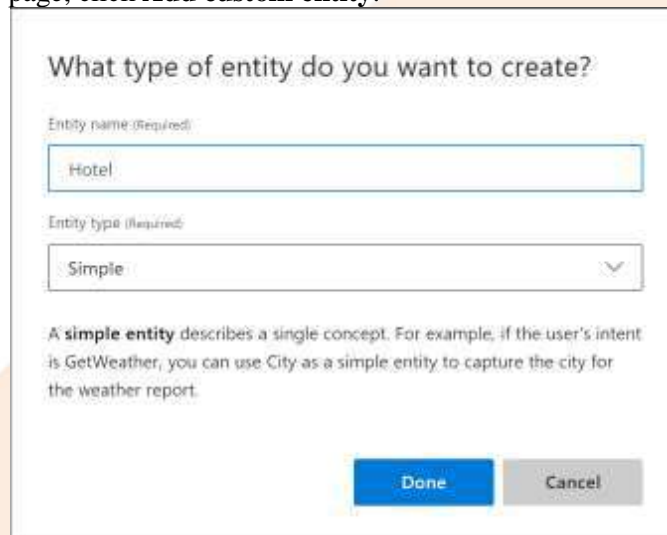


In **Add prebuilt domain entities** dialog box, click the prebuilt entity you want to add (for example, "Places.AbsoluteLocation"). Then click **Done**.



in our Lab we will need to add two more custom entities: the hotel name and the airport code:

3. Open the LuisBot app by clicking its name on **My Apps** page, and then click **Entities** in the left panel.
4. On the **Entities** page, click **Add custom entity**.

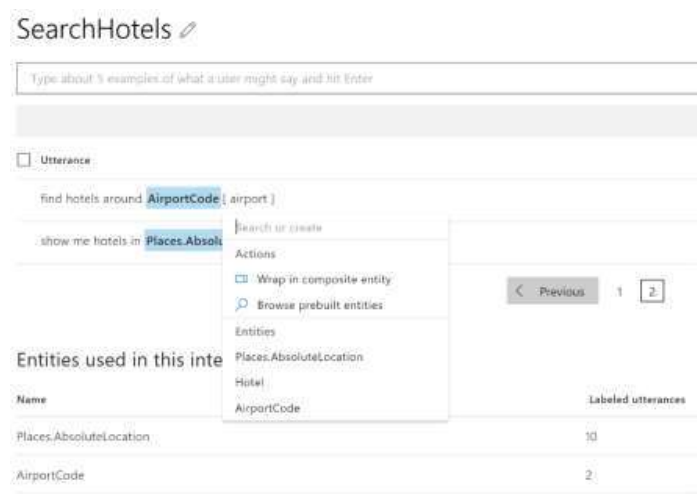


5. In the Add Entity dialog box, type "Hotel" in the Entity name box, select Simple from the **Entity type** list, and then click **Done**.
6. Repeat the previous step to add **AirportCode** entity the the custom entities

TASK 5: LABEL UTTERANCES

After adding intents, utterances and entities, your next step is to label the utterances. Utterances are labeled in terms of intents and entities.

1. On the **intents** page, click on **SearchHotels** intent.
2. Select one of the utterance ,ex “ hotels around CDG airport” and click on CDG, In the entity drop-down box that appears, you can either click an existing entity (if available) to select it, or add a new entity by typing its name in the text box and clicking **Create entity**. Now and because we’ve already created the entities select “**AirportCode**”.



3. repeat the previous step with other utterances, with labeling Cities/Addresses with **Places.AbsoluteLocation**, labeling airport codes with **AirportCode** entity and hotel name with **Hotel** entity.

TASK 6: IMPROVE PERFORMANCE USING FEATURES

You can add features to your LUIS app to improve its performance. Features help LUIS recognize both intents and entities, by providing hints to LUIS that certain words and phrases are part of a category or follow a pattern. When your LUIS app has difficulty identifying an entity, adding a feature and retraining the LUIS app can often help improve the detection of related intents and entities.

- **Phrase list** features contain some or all of an entity's potential values. If LUIS learns how to recognize one member of the category, it can treat the others similarly.
- **Pattern features** help your LUIS app easily recognize regular patterns that are frequently used in your application's domain, such as the pattern of flight numbers in a travel app or product codes in a shopping app.

- click **Features** in your app's left panel.
- On the **Features** page, click **Add phrase list**.
- In the **Add Phrase List** dialog box, type "Show" as the name of the phrase list in the Phrase list name box.

The screenshot shows the 'Edit Phrase list' dialog box. It has a title bar 'Edit Phrase list'. Below it, there are two input fields: 'Name (Required)' with the value 'Show' and 'Value (Required)' with the value 'show,find,look,search,locate,get'. Below these fields, there are two sections: 'Phrase list values' and 'Related Values'. The 'Phrase list values' section shows a list of values: 'show', 'find', 'look', 'search', 'locate', and 'get', each with a small 'x' icon to its right. The 'Related Values' section shows a list of values: 'getting', 'take', 'obtain', 'make', 'obtaining', 'able', 'call', 'delete', 'book', and 'locating', each with a small '+' icon to its right. Above the 'Related Values' section, there are two buttons: 'Add all' and 'Recommend'. Below the 'Phrase list values' section, there is a checkbox labeled 'These values are interchangeable' which is checked. At the bottom of the dialog box, there are two buttons: 'Save' and 'Cancel'.

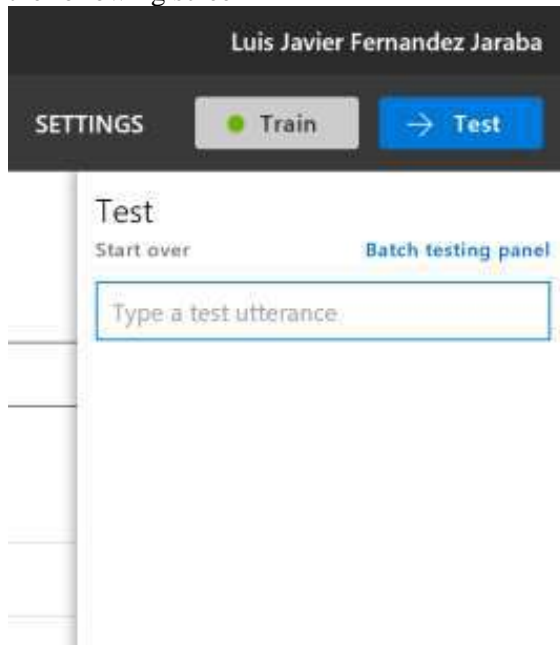
1. In the **Value** box, type the values of the phrase list. You can type one value at a time, or a set of values separated by commas (e.g. show, find, look, search, etc.), and then press Enter.
2. If your app culture is English, LUIS can propose some related values to add to your phrase list. Click **Recommend** to get a group of proposed values that are semantically related to the added value(s). You can click any of the proposed values to add it, or click **Add All** to add them all.
3. Click **Is exchangeable** if the added phrase list values are alternatives that can be used interchangeably.
4. Click **Is active** if you want this phrase list to be active (i.e. applicable and used) in your app.
5. Click **Save**. The phrase list will be added to phrase list features on the **Features** page.

TASK 7: TRAINING AND TESTING YOUR APP

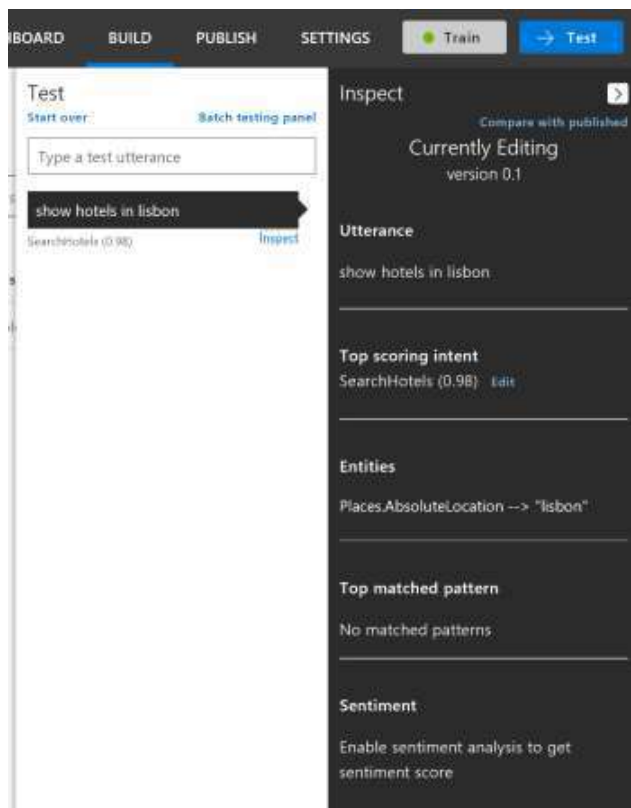
Training is the process of teaching your LUIS app by example to improve its language understanding. If you make updates by adding, editing, or deleting entities, intents, or utterances to your LUIS app, you need to train your app before testing and publishing. When you train a LUIS app, LUIS generalizes from the examples you have labeled, and learns to recognize the relevant intents and entities in the future, which improves its classification accuracy.

To start the iterative process of training, you first need to train your LUIS app at least once.

1. Access your app by clicking its name on **My Apps** page.
2. In your app, click **Train** button to train the LUIS app on the latest updates.
3. If your model is trained, in the right side of the screen, click the **Test** button, it will appear the following screen



4. Write some sentences to search hotels, in the right side of the screen is where your LUIS app returns the test result, which is the predicted interpretation of the utterance



TASK 8: PUBLISH YOUR APP

When you finish building and testing your app, you can publish it as a web service on Azure and get an HTTP endpoint that you can use from your client application.

You can either publish your app directly to the **Production Slot** where end users can access and use your model, or you can publish to a **Staging Slot** where you can iteratively test your app to validate changes before publishing to the production slot.

1. Open your “LuisBot” app by clicking its name on **My Apps** page, and then click **Publish App** in the left panel to access the **Publish App** page. The screenshot below shows the Publish page.

Publish app

Select slot

Staging

Publish

Published version: Slot not published yet

Published date: Application not published

External services settings

☐ Enable Sentiment Analysis

☐ Enable Speech Profiling

Endpoint url settings

Timezone

(GMT) Western Europe Time, London, Lisbon, Casablanca

☒ Include all predicted intent scores

☐ Enable Bing spell checker

Resources and Keys

Add Key

☒ North America Regions ☐ South America Regions ☐ Europe Regions ☐ Asia Regions ☐ Australia Regions

Resource Name	Region	Key String	Endpoint
Starter_Key	western	ta4fca...	https://westus1.api.cognitive.microsoft.com/luis/v2.0/apps/34804355-0500-4006-b2a0-627994615200?subscription-key=ta4fca7de39147a5995e8f1ed9fc4b95&staging=true&verbose=true&timezoneOffset=0

From the **Resources and Keys** list, select an existing endpoint key to assign to the app, or click Add a new key to your account to add a new one.

2. Choose whether to publish to the **Production** or to **Staging** slot by selecting the corresponding value from the **Endpoint Slot** list.
3. If you want the JSON response of your published app to include all intents defined in your app and their prediction scores, click **Include all predicted intent scores Flag** checkbox. Otherwise, it will include only the top scoring intent.
4. Click **Publish**.

If you want to test your published endpoint in a browser using the generated URL, you can click the URL to open it in your browser, then set the URL parameter "&q" to your test query (for example: "&q=show hotels near only airport"), and then press Enter. You will get the JSON response of your HTTP endpoint.

```
{
  "query": "show hotels near only airport",
  "topScoringIntent": {
    "intent": "SearchHotels",
    "score": 0.97505486
  },
  "intents": [
    {
      "intent": "SearchHotels",
      "score": 0.97505486
    },
    {
      "intent": "None",
      "score": 0.0187820084
    },
    {
      "intent": "ShowHotelsReviews",
      "score": 0.00535229733
    },
    {
      "intent": "help",
      "score": 0.00184985274
    }
  ],
  "entities": [
    {
      "entity": "only",
      "type": "AirportCode",
      "startIndex": 17,
      "endIndex": 20,
      "score": 0.9189959
    }
  ]
}
```

Exercise 2: Integrate a Bot with LUIS using NodeJs

Introduction

This exercise will guide you to create a bot application that will solicit the Language Understanding Intelligent Service (LUIS) app created in the exercise 1, and which will try to find the pieces of information that are relevant to user intent.

Objectives

After completing this lab, you will be able to:

- Integrate LUIS with Bot

Prerequisites

Exercise 1

TASK 1: ADD A LUISRECOGNIZER

The BotBuilder Node SDK contains Recognizer plugins that allow to detect intention from user messages using different methods, from Regex to natural language understanding. These Recognizer plugins and the IntentDialog are useful for building more open ended bots that support natural language style understanding.

Go to the module's Labs folder/Begin/Node and open the **LuisBot** Folder with Visual studio code

In **Solution Explorer**, open App.js file and after **builder.UniversalBot** instruction add **LuisRecognizer** that will point at your LUIS model, will listen to every user message and detect intention

```
// You can provide your own model by specifying the 'LUIS_MODEL_URL' environment variable
// This Url can be obtained by uploading or creating your model from the LUIS portal: https://www.luis.ai/
var recognizer = new builder.LuisRecognizer(process.env.LUIS_MODEL_URL);
bot.recognizer(recognizer);
```


TASK 2: CREATE METHODS TO HANDLE INTENTS

Intent recognizers return matches as named intents. To match against an intent from a recognizer you pass the name of the intent you want to handle to **IntentDialog.matches()** or use the dialog's **triggerAction()** by specifying the intent name with **matches** property

- 1) Add method that executes when the **Help** intent is matched.

```
bot.dialog('Help', function (session) {
    session.endDialog('Hi! Try asking me things like \'search hotels in Seattle\', \'search hotels near LAX airport\' or \'show me the reviews of The Bot Resort\');
}).triggerAction({
    matches: 'Help'
});
```

- 2) Add method that executes when the **ShowHotelsReviews** intent is matched.

```
bot.dialog('ShowHotelsReviews', function (session, args) {
    // retrieve hotel name from matched entities
    var hotelEntity = builder.EntityRecognizer.findEntity(args.intent.entities, 'Hotel');
    if (hotelEntity) {
        session.send('Looking for reviews of \'' + hotelEntity.entity + '\', ');
        Store.searchHotelReviews(hotelEntity.entity)
            .then(function (reviews) {
                var message = new builder.Message()
                    .attachmentLayout(builder.AttachmentLayout.carousel)
                    .attachments(reviews.map(reviewAsAttachment));
                session.endDialog(message);
            });
    }
}).triggerAction({
    matches: 'ShowHotelsReviews'
});
```


3) Add method that executes when the **SearchHotels** intent is matched.

```
bot.dialog('SearchHotels', [
  function (session, args, next) {
    session.send('Welcome to the Hotels finder! We are analyzing your
message: \''s\'', session.message.text);
    // try extracting entities
    var cityEntity =
      builder.EntityRecognizer.findEntity(args.intent.entities,
        'builtin.geography.city');
    var airportEntity =
      builder.EntityRecognizer.findEntity(args.intent.entities,
'AirportCode');
    if (cityEntity) {
      // city entity detected, continue to next step
      session.dialogData.searchType = 'city';
      next({ response: cityEntity.entity });
    } else if (airportEntity) {
      // airport entity detected, continue to next step
      session.dialogData.searchType = 'airport';
      next({ response: airportEntity.entity });
    } else {
      // no entities detected, ask user for a destination
      builder.Prompts.text(session, 'Please enter your destination');
    }
  },
  function (session, results) {
    var destination = results.response;
    var message = 'Looking for hotels';
    if (session.dialogData.searchType === 'airport') {
      message += ' near 's airport...';
    } else {
      message += ' in 's...';
    }
    session.send(message, destination);
    // Async search
    Store
      .searchHotels(destination)
      .then(function (hotels) {
        // args
        session.send('I found %d hotels:', hotels.length);
        var message = new builder.Message()
          .attachmentLayout(builder.AttachmentLayout.carousel)
          .attachements(hotels.map(hotelAsAttachment));
      });
  }
]);
```

```
        session.send(message);  
        // End  
        session.endDialog();  
    });  
}  
]).triggerAction({  
    matches: 'SearchHotels',  
    onInterrupted: function (session) {  
        session.send('Please provide a destination');  
    }  
});
```

TASK 3: UPDATE THE LUIS MODEL URL

- 4) Open the file “.env” and set the LUIS_MODEL_URL value with your LUIS app url:

This Url can be obtained by uploading or creating your model from the LUIS portal: <https://www.luis.ai/>
LUIS_MODEL_URL=

TASK 4: TRY YOUR BOT

- 1) Start your application in debug or release mode.
- 2) Copy the URL of your application
- 3) Launch the Bot framework emulator

LuisBot

A sample bot using LuisDialog to integrate with a LUIS.ai application.

Visit [Bot Framework](#) to register your bot. When you register it, remember to set your bot's endpoint to

https://your_bot_hostname/api/messages

- 4) Paste the URL and add **/api/messages** as a suffix.
- 5) You will see the following in the Bot Framework Emulator when opening and running the sample solution.

