



800 XP

Add conversational intelligence to your apps by using Language Understanding Intelligent Service (LUIS)

54 min • Module • 7 Units

★★★★☆ 4.6 (1,296) [Rate it](#)[Intermediate](#) [Developer](#) [Solutions Architect](#) [AI Engineer](#) [Student](#) [Azure](#) [SDKs](#) [Bot Service](#) [Cognitive Services](#)[Language Understanding](#)

In this module, we'll introduce you to [Language Understanding Intelligent Service \(LUIS\)](#) and show how to create a LUIS application.

In this module, you'll:

- Learn what LUIS is.
- Learn about key features of language understanding, such as intents and utterances.
- Build and publish a natural-language machine-learning model.

[Start >](#) [Bookmark](#) [Add to collection](#)

Prerequisites

None

This module is part of these learning paths

[Evaluate text with Azure Cognitive Language Services](#)

Introduction

1 min

Overview of Language Understanding Intelligent Service (LUIS)

4 min

Create a LUIS service resource

10 min

Exercise - Add intents and entities

15 min

Train the LUIS model

15 min

Knowledge check

7 min

Summary

2 min

Summary

Introduction	3
Learning objectives	3
Overview of Language Understanding Intelligent Service (LUIS)	4
Create a LUIS service resource.....	6
Create a LUIS app	10
Exercise - Add intents and entities.....	12
Create entities.....	15
Map search subjects to the facet entity	16
Train the LUIS model	20
Create a public endpoint for the LUIS service.....	20
Publish the app.....	23
Knowledge check	27
Check your knowledge	27
Summary.....	28
Cleanup	28

Introduction

A key factor in AI applications is the ability to interact intelligently with the user. Some of the most common AI interactions occur through bots. You can build conversational intelligence into your bot by using Azure Language Understanding Intelligent Service (LUIS).

LUIS is an Azure Cognitive Services API that applies custom machine-learning intelligence to a user's conversational natural-language text. LUIS uses certain aspects of the text to predict the user's overall meaning and pull out relevant detailed information. Your applications can use this information to interact with the user.

Note

This module requires an Azure subscription. The services you create and use are free, but you'll need an active subscription or trial to complete the exercises. If you don't have an Azure subscription, [create a free account](#) before you begin.

Learning objectives

In this module, you'll:

- Gain an understanding of what LUIS is.
- Learn about key features of LUIS, such as intents and utterances.
- Build and publish a LUIS model.

Overview of Language Understanding Intelligent Service (LUIS)

Understanding language is something that even humans get wrong from time to time. A good example is the use of slang terms or localized phrases. Suppose you're in Indonesia at a public place, perhaps a mall or a restaurant, and you're searching for the restroom. Your Indonesian language lessons might have taught you to use the phrase, "*Di mana kamar kecil?*"

While this phrase is technically correct, it applies mainly to seeking the restroom in someone's house. "*Kamar kecil*" literally means "small" (*kecil*) "room" (*kamar*). In public, it's more correct to ask, "*Di mana WC?*", or "*Di mana toilette?*" But if you use "*Di mana kamar kecil?*" in public, almost all Indonesians will know what you mean.

But what happens if you ask a computer that question? Will it give you the correct answer? Or will it direct you to a "small room" that isn't a restroom?

Likewise, in English, a native English speaker might understand a phrase whose meaning isn't clear to a nonnative speaker, or even to some native English speakers. How many people understand the phrase "swing the door to"? It means the same thing as "shut the door" or "close the door," but it might not be clear to everyone.

For AI to understand language, specific aspects are critical to aiding the algorithm in making comparisons and distinctions. This area is where Language Understanding Intelligent Service (LUIS) comes in.

LUIS makes use of three key aspects for understanding language:

- **Utterances:** An utterance is input from the user that your app needs to interpret.
- **Intents:** An intent represents a task or action the user wants to do. It's a purpose or goal expressed in a user's utterance.
- **Entities:** An entity represents a word or phrase inside the utterance that you want to extract.

Here's an example. Suppose you're using LUIS in a bot to help a user book a flight. A user may use the following utterance, "Book 2 tickets on a flight to New York for New Year's Eve." If we evaluate this utterance for key aspects, we can

determine the user's intent. The user wants to book a flight. We can state the intent as *BookFlight*.

Entities aren't only words or phrases, but also simply data. This data helps provide specific context for the utterance and aids the algorithm in more accurately identifying the intent. Not every utterance contains entities, though.

In the sample utterance used earlier, we can identify entities like:

- **New York:** We can classify this entity as *Location.Destination*.
- **New Year's Eve:** We can classify this entity as *Event*.
- **The number 2:** This number maps to a built-in entity. In LUIS, such an entity is known as a *prebuilt entity*, specifically a prebuilt number.

With this understanding, let's look at how to create a LUIS app.

Create a LUIS service resource

Let's look at how we can use LUIS to add natural language capabilities to a picture-management bot. LUIS allows you to map natural language *utterances* to *intents*. In other words, LUIS maps the user's words, phrases, or sentences to tasks or actions the user wants to do.

Our application might have several intents: finding pictures, sharing pictures, and ordering prints of pictures. We'll give LUIS a few example utterances of ways users might ask for each of these things. LUIS will then map additional new utterances to each intent based on what it learns over time.

Note

This is an optional exercise. To use a LUIS service that's tied to your Azure account, follow these steps to create the LUIS service. To use a test environment only, go to the "Create a LUIS app" exercise later on this page.

1. Sign in to the [Azure portal](#).
2. In the left pane, select + **Create a resource**.
3. In the **Search the Marketplace** box, type **LUIS**, and then press Enter.
4. In the search results, select **Language Understanding**.
5. Select **Create**.



6. Leave the Create options set to **Both**
7. Enter a unique name for your LUIS service.
8. Choose a subscription.
9. Create a new resource group named **LearnRG**.
10. For **Authoring Location**, choose the one nearest you.
11. For **Authoring pricing tier**, select **F0**
12. Set your **Prediction location** to the same region you chose for **Authoring location**
13. For **Prediction pricing tier**, select **F0**.
14. Select **Create**.

[Home](#) > [New](#) > [Marketplace](#) > [Language Understanding](#) >

Create

Cognitive Services

[* Basics](#) [Tags](#) [Review + create](#)

Language Understanding (LUIS) is a natural language processing service that enables you to understand human language in your own application, website, chatbot, IoT device, and more. After you configure and publish your LUIS model, your application can easily receive user input in natural language and take action. You don't need to understand machine learning to solve the problem of extracting meaning from input. Instead you get to focus on your own application logic and let LUIS do the heavy lifting on your behalf. After your LUIS model is built and deployed, it exports a simple HTTP endpoint that is called by your application. [Learn more](#)

Create options

Both

Authoring

Prediction

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *

Azure Pass – Sponsorship



Resource group *

Prep-AI100-DougLuna

[Create new](#)

Name *

PrepAI100LUIS



Authoring Resource

Select pricing and location for Authoring Resource

Authoring location *

(US) West US



Authoring pricing tier *

Free F0 (5 Calls per second, 1M Calls per month)



Prediction Resource

Select pricing and location for Prediction Resource

Prediction location *

(US) West US



Prediction pricing tier *

Free F0 (5 Calls per second, 10K Calls per month)

**Review + create**

Next : Tags >

Microsoft Azure

Search resources, ser

[Home](#) > [New](#) > [Marketplace](#) > [Language Understanding](#) >

Create

Cognitive Services

[* Basics](#) [Tags](#) [Review + create](#)

Project details

Subscription	Azure Pass – Sponsorship
Resource group	Prep-AI100-DougLuna
Name	PrepAI100LUIS
Authoring location	West US
Authoring pricing tier	F0
Prediction location	West US
Prediction pricing tier	F0

Create

< Previous : Tags

Automation options

After the deployment has finished, go the resource page for the service. You'll need one of the displayed subscription keys for later exercises. You can either use the "Grab your keys" option in the **Quick start** section, or switch to the **Keys** section to see the two created keys.

Microsoft Azure

Search resources, services, and docs (G+/)

[Home](#) > [Microsoft.CognitiveServices.LUISAllInOne](#) | Overview >

PrepAI100LUIS-Authoring | Keys and Endpoint

Cognitive Services

<< [Regenerate Key1](#) [Regenerate Key2](#)

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

RESOURCE MANAGEMENT

Quick start

Keys and Endpoint

Pricing tier

Networking

Identity

Billing By Subscription

Properties

These keys are used to access your Cognitive Service API. Do not share your keys. Store them securely– for example, using Azure Key Vault. We also recommend regenerating these keys regularly. Only one key is necessary to make an API call. When regenerating the first key, you can use the second key for continued access to the service.

Show Keys

KEY 1

.....

KEY 2

.....

ENDPOINT

https://prepai100luis-authoring.cognitiveservices.azure.com/

LOCATION ⓘ

westus

Create a LUIS app

Azure requires you to create the LUIS app in the same geographic location where you created the service. If you didn't create the service in the previous optional exercise, use the location closest to you.

1. Select one of the following URLs, and open it in a new browser window.
 - **North America:** <https://www.luis.ai/>
 - **Europe:** <https://eu.luis.ai/>
 - **Australia:** <https://au.luis.ai/>

Note

There are three LUIS websites, based on region. You must author and publish in the same region. Select the closest region to you.

2. Select the **Sign in** link in the upper-right corner.
3. Sign in with your Microsoft account.
4. On the **Welcome** page, select **Create a LUIS app now**.
5. Select **+ Create a new app**.
6. Give your LUIS app a name, for example, **PictureBotLUIS**.
7. For **Culture**, select the appropriate choice.
8. Give your LUIS app a description so it's clear what the app's purpose is.
9. Select **Done**.

The screenshot shows the 'My Apps' interface with a 'Create new app' dialog box open. The background interface includes a 'Subscription' dropdown set to 'Azure Pass – Sponsorship', an 'Authoring resource' dropdown set to 'PrepAI100LUIS-Authoring', and a search bar labeled 'Search apps ...'. The dialog box contains the following fields:

- Name ***: A text input field containing 'PictureBotLUIS'.
- Culture ***: A dropdown menu set to 'English'. Below it is a note: 'Note: Culture is the language that your app understands and speaks, not the interface language.'
- Description**: A text input field containing 'MS-Learn Picture Bot LUIS'.
- Prediction resource (Optional)**: A dropdown menu set to 'PrepAI100LUIS'. Below it is a note: 'Note: Only resources allowed for use in this region are displayed.'

At the bottom right of the dialog are two buttons: 'Done' (in blue) and 'Cancel'.

The newly created app will open to the Dashboard. Select Build on the top nav bar to switch to the Build view for editing your app.

Exercise - Add intents and entities

Our next step for the PictureBotLUIS app is creating intents that map to user requests. We want our picture-management bot to understand how to:

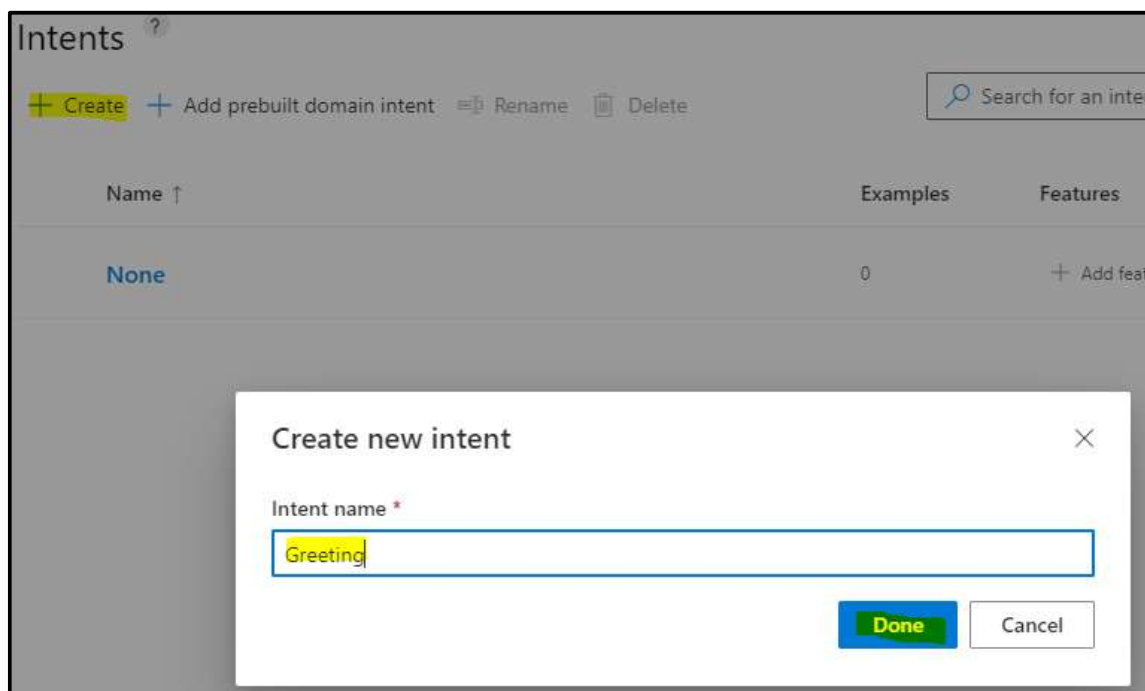
- Find pictures.
- Share pictures on social media.
- Order prints of pictures.
- Greet the user.

Let's create intents for each of these items.

Note

There is one intent already present named **None**. Random utterances that don't map to any of your intents can be mapped to this intent.

1. Select **+ Create new intent**.
2. Name the first intent **Greeting**, and then select **Done**.



Because our scenario for this application is to integrate with a bot, you'll provide examples of utterances that users might say when greeting the bot initially.

3. Enter a greeting utterance, such as **hello**, and press Enter.

4. Repeat the previous step to create values for each of the following utterances: **hi, hola, yo, hey, greetings**.


Tip

You should always provide at least five example utterances for each intent.








5. Your utterances for the **Greeting** intent should look similar to the following image.






Greeting

Machine learning features

 Add feature

Examples

 Confirm all entities  Move to  Delete ...   View options  

Example user input	Score
<input type="text" value="Type an example of what a user might say and hit Enter."/>	
ola	
hola	
yo	
greetings	
hey	

1. Create another intent named **SearchPics**.
2. Add the following values as utterances for the **SearchPics** intent:
 - **find outdoor pics**
 - **are there pictures of a train?**
 - **find pictures of food**
 - **search for photos of boys playing**
 - **give me colorful pictures**
 - **show me beach pics**
 - **I want to find dog photos**
 - **find pictures of German shepherds**
 - **search for pictures of men indoors**
 - **show me pictures of men wearing glasses**
 - **I want to see pics of smiling people**
 - **show me baby pics**

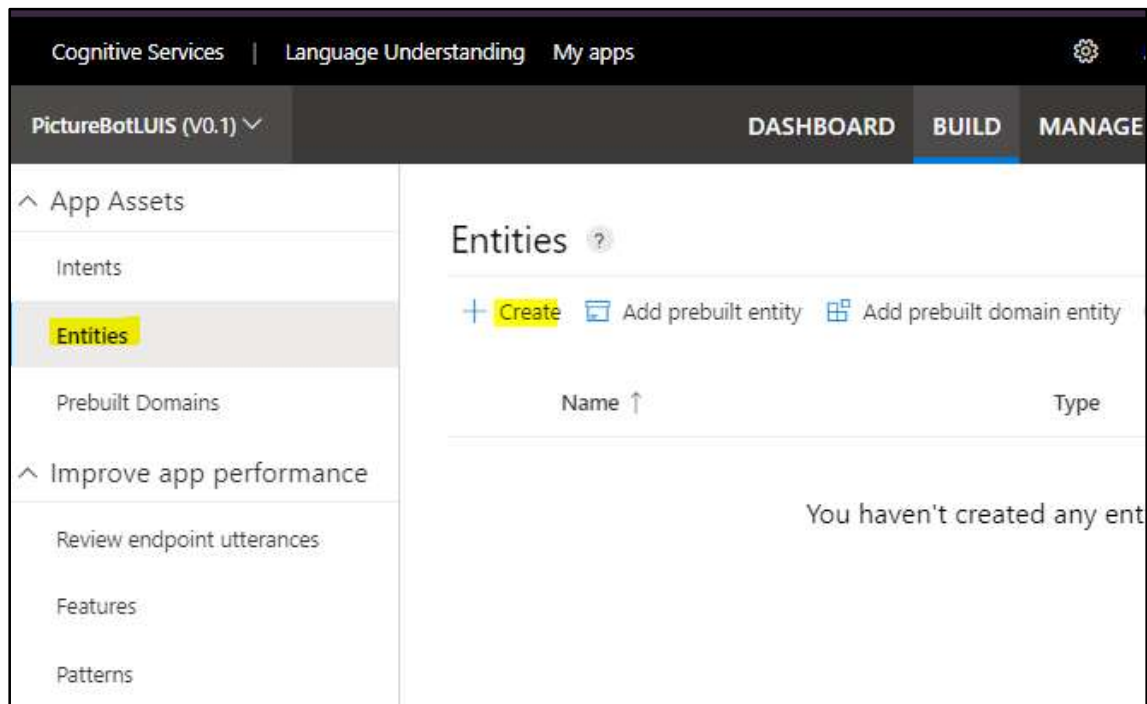
The screenshot shows the Microsoft Bot Framework LUIS (Language Understanding Intelligent Service) interface. The top navigation bar includes 'PictureBotLUIS (V0.1)', 'DASHBOARD', 'BUILD' (active), and 'MANAGE'. On the right, there are 'Train' and 'Publish' buttons. The left sidebar shows 'App Assets' with 'Intents' selected, and 'Improve app performance' with 'Review endpoint utterances', 'Features', and 'Patterns'. The main area is titled 'SearchPics' and shows 'Machine learning features' with an 'Add feature' button. Below this is the 'Examples' section, which contains a table of user input examples and their scores. The table has two columns: 'Example user input' and 'Score'. The examples listed are: 'show me pics of robots', 'show me baby pics', 'i want to see pics of smiling people', 'show me pictures of men wearing glasses', and 'search for pictures of men indoors'. Each example has a score of 1.0. There is also a text input field for adding new examples, with a placeholder text: 'Type an example of what a user might say and hit Enter.'

Example user input	Score
show me pics of robots	1.0
show me baby pics	1.0
i want to see pics of smiling people	1.0
show me pictures of men wearing glasses	1.0
search for pictures of men indoors	1.0

Create entities

Next, let's create the entities we need to capture specific requests from users. For example, when users want to search the pictures, they might specify what they're looking for.

1. In the left column, select **Entities**, and then select **+ Create**.



2. Name the entity **facet** (to represent one way to identify an image).
3. Select **Machine learned** for **Entity type**. Then select **Create**.

Create an entity

Name *

facet

Type:

☒ Machine learned

☐ List

☐ Regex

☐ Pattern.any

Machine learned entities are learned from context. Use an ML entity to identify data that are not always well-formatted but have the same meaning. An ML entity can be composed of smaller subentities, each of which can have its own properties.

Example

[Hide subentities](#)

"Book 2 adult business tickets to Arrabury Airport"

Number PassengerClass TravelClass

TicketOrder

"TicketOrder" has components "Number," "PassengerClass", and "TravelClass".

☐ Add structure

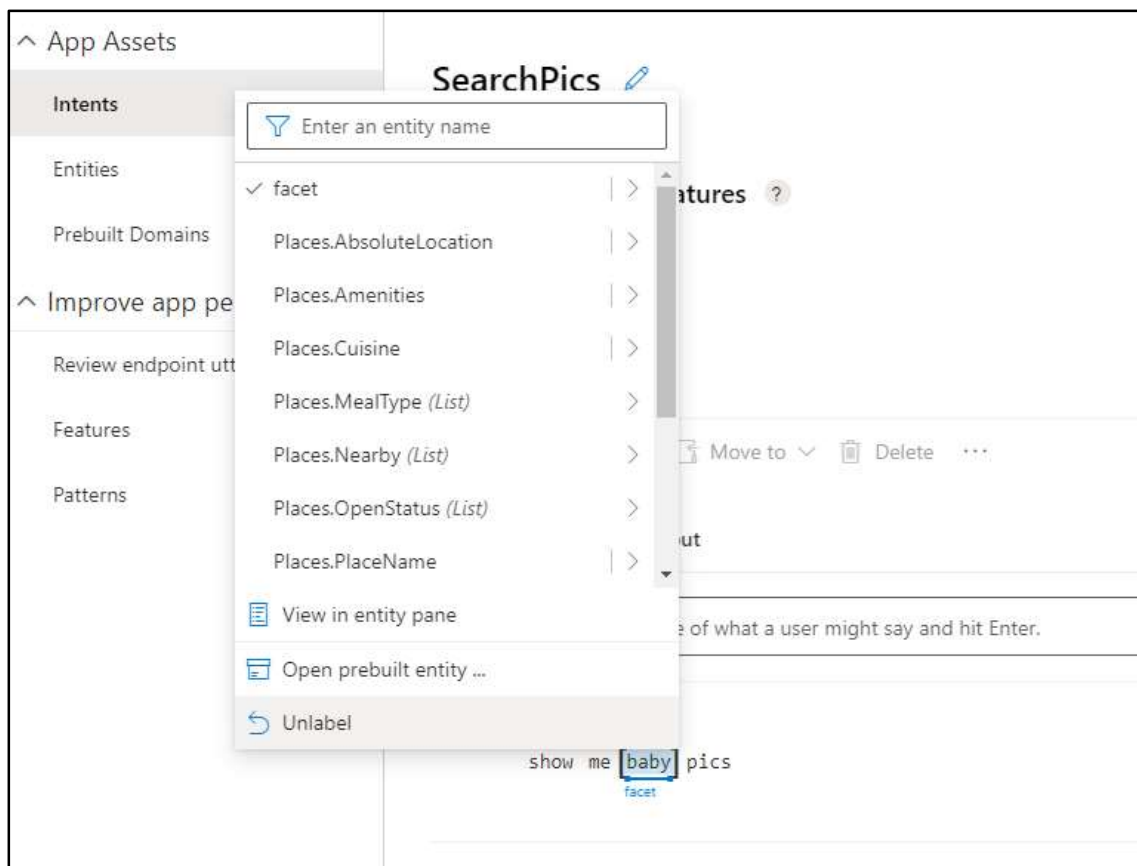
Create

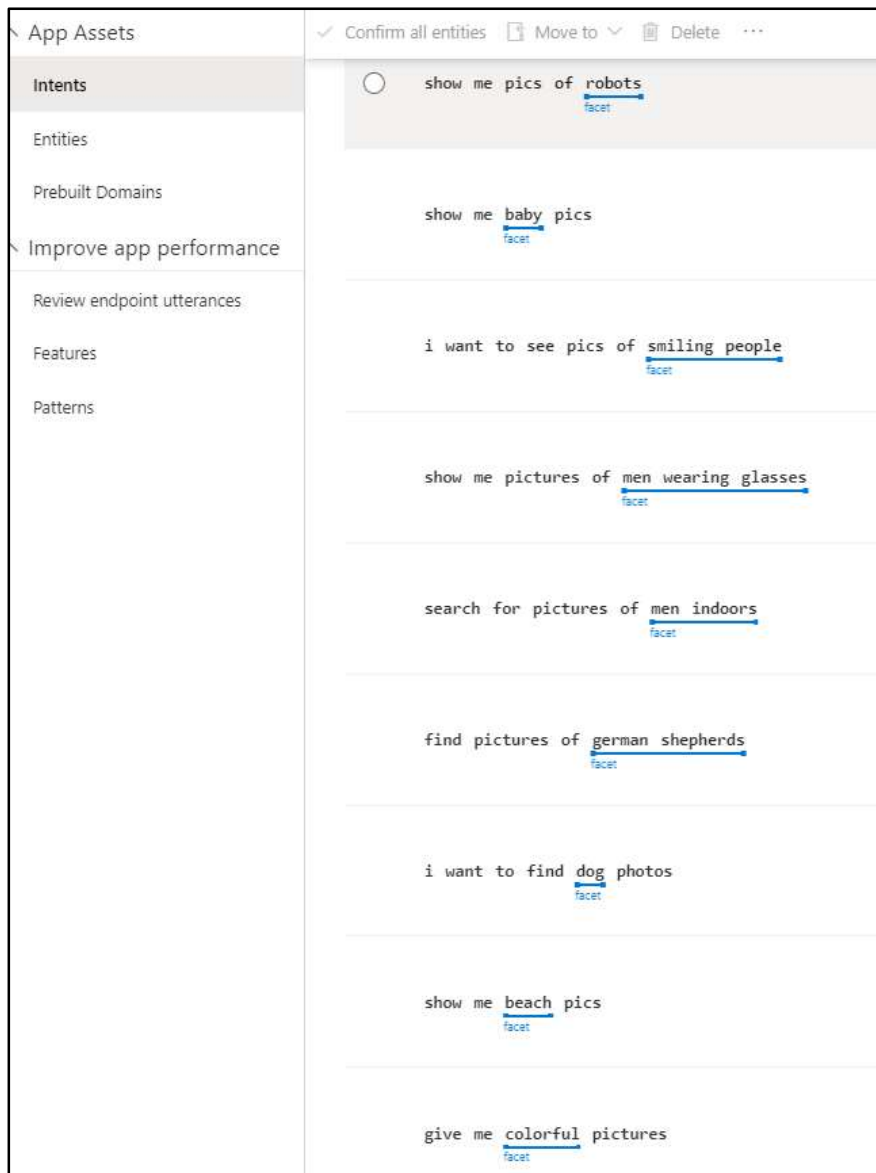
Cancel

Map search subjects to the facet entity

Next, we have to teach LUIS how to pick out the search subject as the **facet** entity. Whatever the **facet** entity picks up is what the app will search for.

1. Switch back to the **Intents** page and select the **SearchPics** intent.
2. Hover over the utterance and click the keyword that specifies the *search subject*, and then select the **facet** entity. For example, if the utterance is "show me baby pics", the subject would be "baby".





1. Add two more intents with related utterances, as shown in the following table:

Intent name	Utterances
SharePic	Share this pic, Can you tweet that?, Post to Twitter
OrderPic	Print this picture, I would like to order prints, Can I get an 8x10 of that one?, Order wallets

PictureBotLUIS (V0.1) ▾


App Assets

- Intents
- Entities
- Prebuilt Domains


Improve app performance



- Review endpoint utterances
- Features
- Patterns

SharePic

Machine learning features 

+ Add feature

Examples 

✓ Confirm all entities  Move to ▾  Delete ...

Example user input

Type an example of what a user might say and hit Enter.

share this pic , can you tweet that ? , post to twitter

PictureBotLUIS (V0.1) ▾


App Assets

- Intents
- Entities
- Prebuilt Domains


Improve app performance



- Review endpoint utterances
- Features
- Patterns

OrderPic

Machine learning features 

+ Add feature

Examples 

✓ Confirm all entities  Move to ▾  Delete ...

Example user input

Type an example of what a user might say and hit Enter.

print this picture , i would like to order prints , can i get an 8x10 of that one ? , order wallets

2. To finish out the exercise, add some utterances to the existing **None** intent. Make sure these utterances don't match the context of this LUIS app. Some examples are:
- **I'm hungry for pizza**
 - **Search videos**
 - **Show me how to drive a car**

^ App Assets

Intents

Entities

Prebuilt Domains

^ Improve app performance

Review endpoint utterances

Features

Patterns

None

Machine learning features ?

+ Add feature

Examples ?

✓ Confirm all entities  Move to ▾  Delete ...

Example user input

Type an example of what a user might say and hit Enter.

talk to me

show me how to drive a car

search videos

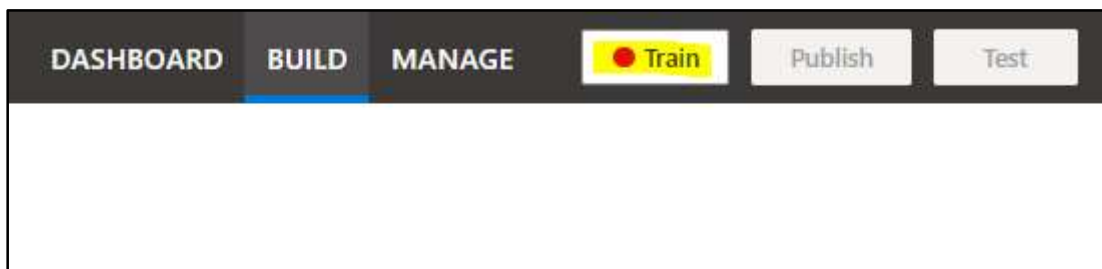
i ' m hungry for pizza

Train the LUIS model

- 15 minutes

We're now ready to train our model. In this exercise, you will perform a simple training operation in order to test your model. The testing will take place using the built-in testing panel in the LUIS portal.

1. In the top bar, select **Train**. During training, LUIS builds a model to map utterances to intents based on the training data you've provided.



Tip

Training is not always immediate. Sometimes it gets queued and can take several minutes.

Create a public endpoint for the LUIS service

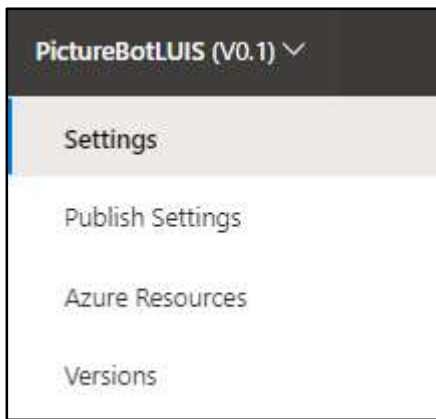
1. After training is finished, select **Manage** in the top toolbar. The following options will appear on the left toolbar:

Note

The categories on the left pane may change as the portals are updated. As a result, the keys and endpoints may fall under a different category than the one listed here.

- **Publish settings**
- **Azure Resources**
- **Versions**
- **Collaborators**

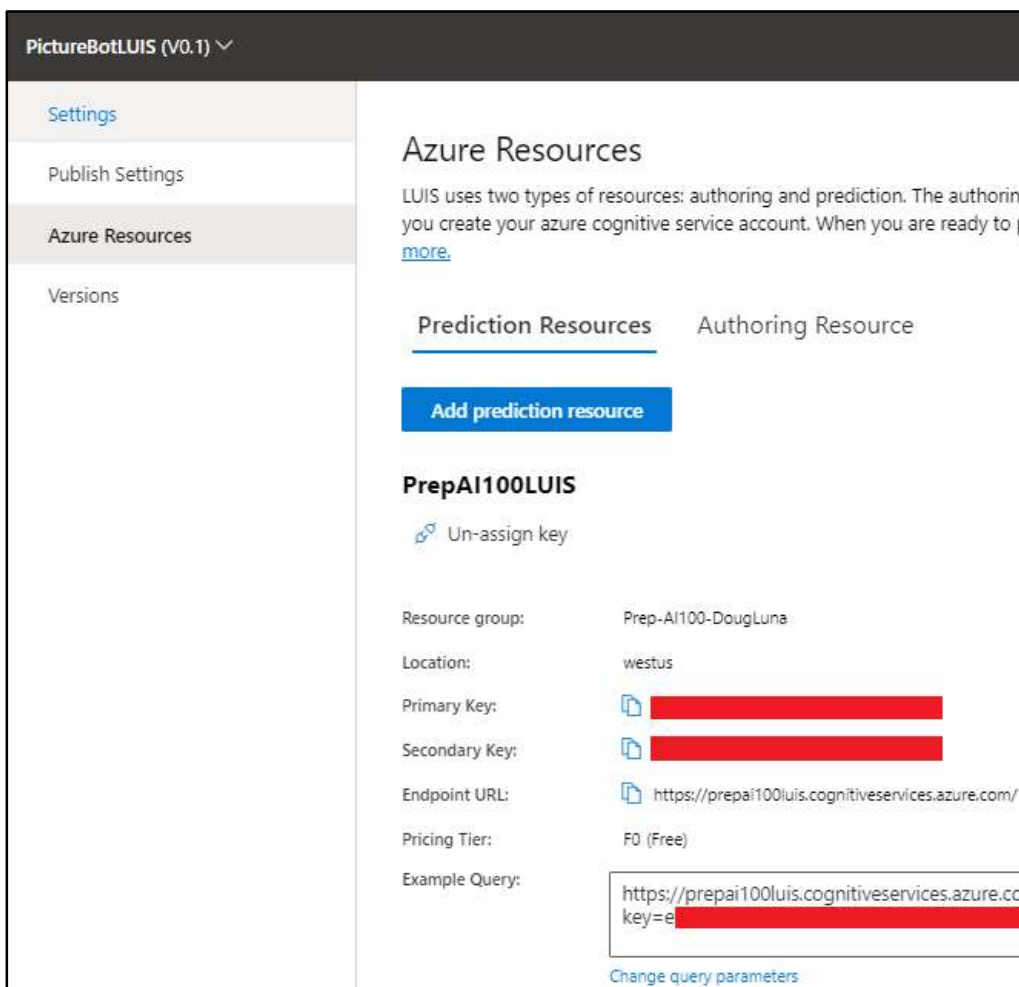




2. Select **Azure Resources**. This screen is used to manage the URL endpoints used to access the LUIS service.

Note

An endpoint named **Starter_Key** is automatically created for testing purposes, and you could use that here - however to use the service in a production environment or inside of an application, you will always want to tie it to a real Language Understanding resource created in Azure.



3. You should see a **Prediction Resource** and a **Starter_Key** resource already created. If you see the **Prediction Resource**, advance to the next section on **Publish the app**.
4. If you do not see an existing **Prediction Resource**, select **Add prediction resource**. The **Tenant** will already be selected.
5. Select your subscription, and the resource you created in the Azure portal earlier and then select **Done** to connect the Language Understanding resource to the LUIS service.

Azure Resources

LUIS uses two types of resources: authoring and prediction. The authoring key is needed for authoring your LUIS app. The prediction key is needed for publishing your LUIS app. [Learn more](#)

Prediction Resources Authoring Resource

Add prediction resource

Add a prediction resource

Your application needs a Language Understanding resource for you to access your language model. [Learn more about LUIS resources in Azure](#)

Azure directory ?

FMU - Faculdades Met Un Ed Ltda. - 7850349 - Rede Internacional de Universidades ...

Subscription *

Azure Pass – Sponsorship

LUIS resource*

MS-Learn

Pricing tier: Standard (S0)

Managed identity: Disabled

[Create a new LUIS prediction resource?](#)

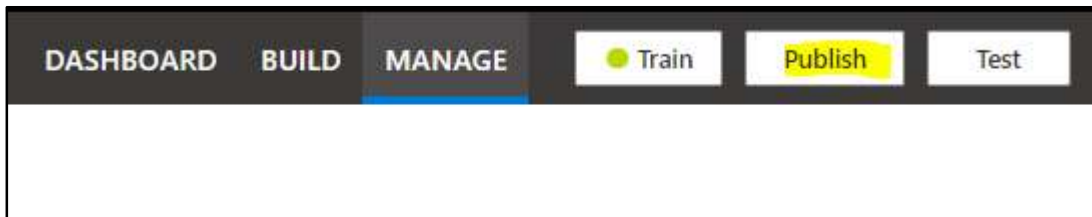
Done Cancel

Publish the app

1. In the top toolbar, select **Publish**.

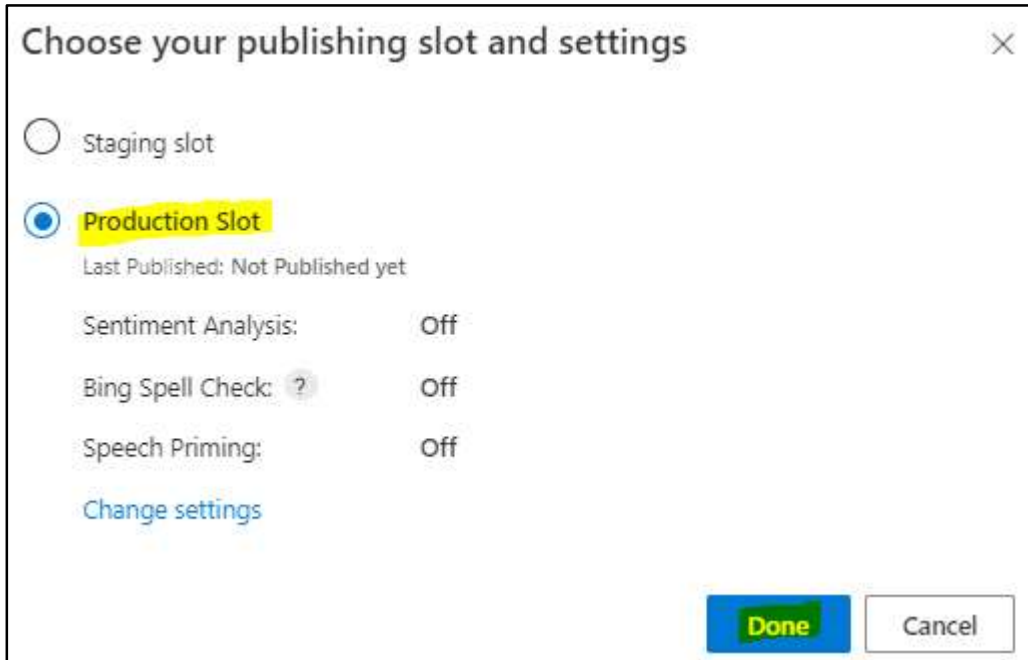
Note

You can publish to your **Production** or **Staging** endpoint. Select **Production**, and read about the reasons for the two endpoints.



2. Under **Choose your publishing slot and settings**, select **Production Slot** and then select **Done**.

Publishing creates an endpoint to call the LUIS model. The endpoint URL will be displayed. Copy the endpoint URL and add it to your list of keys for future use.



3. In the top bar, select **Test**. Try typing a few utterances and see which intents are returned. Here are some examples you can try:



Utterance	Result	Score meaning
Show me pictures of a local beach	Returns the SearchPics intent with a score	Positive match
Hello	Returns the Greeting intent with a score	Fairly positive match
Send to Tom	Returns the Utilities with a low score	Needs retraining or doesn't match any intents

Test

[Start over](#) [Batch testing panel](#)

show me pictures of local beach

SearchPics (0.957) [Inspect](#)

Version: 0.1

[Start over](#) [Compare with published](#)

User input

show me pictures of local beach

Top-scoring intent

SearchPics (0.957) [Add to example utterances](#) ▾

ML entities ☐ Debug required features ?

facet

local beach

Test

[Start over](#) [Batch testing panel](#)

hello

Greeting (0.966) [Inspect](#)

show me pictures of local beach

SearchPics (0.957) [Inspect](#)

Version: 0.1

[Start over](#) [Compare with published](#)

User input

hello

Top-scoring intent

Greeting (0.966) [Add to example utterances](#) ▾

ML entities ☐ Debug required features ?

No predictions

Test

[Start over](#) [Batch testing panel](#)

send to tom

None (0.136) [Inspect](#)

hello

Greeting (0.966) [Inspect](#)

show me pictures of local beach

SearchPics (0.957) [Inspect](#)

Version: 0.1

[Start over](#) [Compare with published](#)

User input

send to tom

Top-scoring intent

None (0.136) [Add to example utterances](#) ▼

ML entities

☐ Debug required features ?

No predictions

To retrain the model for utterances with low scores, take the following steps:

1. Beside the low-scoring utterance (in this case, **Send to Tom**), select **Inspect**.

Test

[Start over](#) [Batch testing panel](#)

send to tom

None (0.136) [Inspect](#)

2. Beside **Top-scoring intent**, select the drop-down and choose **SharePic** from the list.

Test

[Start over](#) [Batch testing panel](#)

send to tom

None (0.136) [Inspect](#)

hello

Greeting (0.966) [Inspect](#)

show me pictures of local beach

SearchPics (0.957) [Inspect](#)

Version: 0.1

[Start over](#) [Compare with published](#)

User input

send to tom

Top-scoring intent

None (0.136) [Add to example utterances](#) ▼

Greeting (0.082)
None (0.136)
OrderPic (0.001)
SearchPics (0.113)
SharePic (0.004)

ML entities

☐ Debug required features ?

No predictions

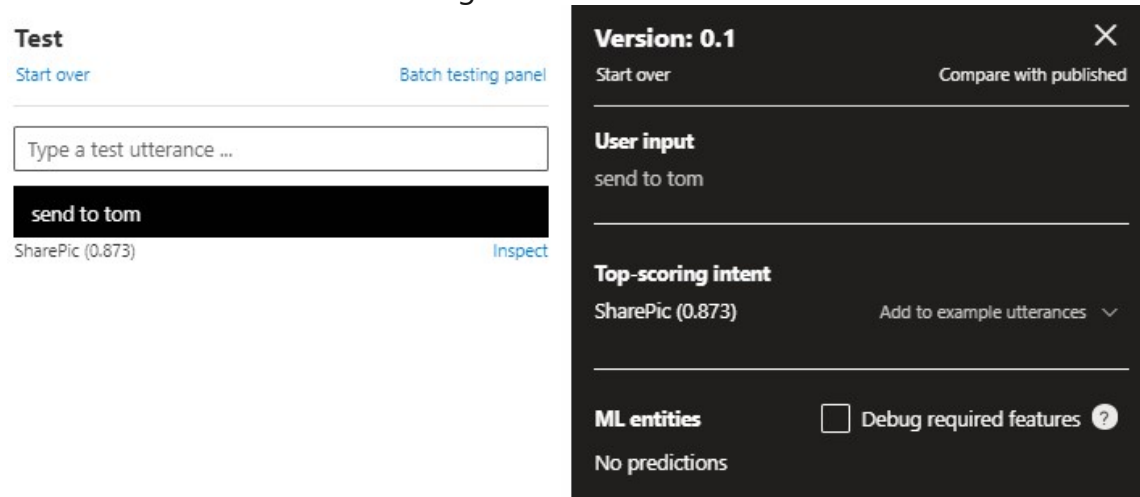
Composite entities

No predictions

3. Close the **Test** panel.
4. Select the **Train** button to retrain your model.



5. Test the **Send to Tom** utterance again. It should now return the **SharePic** intent with a higher score.



Your LUIS app is now ready to be used by client applications, tested in a browser through the listed endpoint, or integrated into a bot.

Knowledge check

Check your knowledge

1. In a LUIS app, what's the purpose of an utterance?

- ☐ An utterance represents a task or action that the user wants to do.
- ☐ Utterances are input from the user that your app needs to interpret.
- ☐ An utterance represents a word or phrase inside an intent.

2. What is the purpose of the None intent?

- ☐ The None intent is used when a user sends null requests.
- ☐ The None intent is used to remove spaces from the beginnings and endings of utterances.
- ☐ Random utterances that don't map to any of your intents can be mapped to None.

Knowledge check

7 minutes

✓ 200 XP

Check your knowledge

1. In a LUIS app, what's the purpose of an utterance?

☐ An utterance represents a task or action that the user wants to do.

☒ Utterances are input from the user that your app needs to interpret. ✓

Correct. An utterance is a phrase a user might use to interact with the application, such as, "Book a flight to New York."

☐ An utterance represents a word or phrase inside an intent.

2. What is the purpose of the None intent?

☐ The None intent is used when a user sends null requests.

☐ The None intent is used to remove spaces from the beginnings and endings of utterances.

☒ Random utterances that don't map to any of your intents can be mapped to None. ✓

Correct. The None intent serves as a catchall for utterances that don't map to existing intents.

Summary

Language Understanding Intelligent Service (LUIS) allows your application to understand what people want in their own words. With LUIS, you can build applications that receive user input in natural language and extract meaning from it.

In this module, you created a simple LUIS application. You then added some intents, utterances, and entities to test how it interacts with a user. The idea behind building this LUIS application was to integrate it into a bot application. To integrate LUIS into your own AI applications, use the API documentation to understand how to do that integration.

Cleanup

1. To avoid any unexpected costs in your Azure account, delete the **LearnRG** resource group. Deleting this group will remove all the resources we created in this module. Here are the steps you need to take:
2. Sign in to the [Azure portal](#).
3. In the left pane, select **Resource groups**, and then find the **LearnRG** resource group.
4. Select the resource group, and either right-click the row or use the **ellipsis** (...) button to open the context menu.
5. Select **Delete resource group**.

6. Enter the name of the **LearnRG** resource group, and then select **Delete**.
Azure will remove all the resources for you.