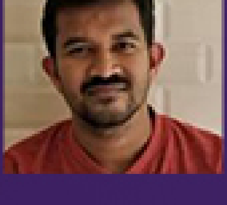
The Implementation & Handling of Multiple Microsoft Chatbot Modules



Author

Rajeesh Menoth

C#Corner

Table of Contents

About the Author

Rajeesh Menoth is a Microsoft Azure Certified Professional and holds a Master's Degree in MBA-Information System Management & a Bachelor's Degree in B.Tech-Information Technology. He is a Software engineer currently working in a multinational company in Bengaluru, India. He has over 7 years of extensive experience with the Microsoft technologies stack, including C#, ASP.NET (Core & MVC) and SQL Server, and Artificial Intelligence experience in Chatbot development using Microsoft Bot Framework, QnA Maker, and LUIS. Rajeesh is a technical author and loves to contribute to the open source community. He writes articles for multiple platforms, including C# Corner, Dzone, and Microsoft TechNet Wiki. For his dedicated contribution to the developer's community, he has been recognized as a C# Corner MVP, Dzone MVB, won a Top Contributor award and TechNet Wiki Guru award from Microsoft. He runs a blog at

https://rajeeshmenoth.wordpress.com/

You can also follow him on

- Facebook : https://www.facebook.com/rajeeshmenoth
- Twitter : https://twitter.com/rajeeshmenoth
- LinkedIn : https://www.linkedin.com/in/rajeeshmenoth/
- Github : https://github.com/rajeeshmenoth

Introduction

Recent trends show that chatbots are essential everywhere, whether it's an entertainment, medical, or knowledge-based website. Since this is an emerging technology, developers all around the world will be wanting to know more about this, specifically, what is happening in the background of chat bot applications.

This book teaches you how to handle and implement multiple chatbot modules in real-world scenarios.

There are important bot modules like QnA and Luis, which are key players in the chatbot e2e (end-to-end) communication or conversation flow. Throughout this book, you will learn to build and handle different types of chatbot modules from beginning to end.

Target Audience

The book is mainly focusing on people who are interested in chatbot module integration in real-time, such as students, teachers and programmers, with and without knowledge of NLP and AI. A basic knowledge of programming and cloud technologies, like Azure, is required. This will help to correlate situation handling in the real-time applications. The language preference is ASP.NET, C#, and Node.

Azure Subscription

A valid Azure subscription key is required for the creation of any of the services in the Azure portal. We can create an Azure subscription in multiple ways, either a free or paid service. Microsoft provides a free Azure subscription for the first 12 months for a single credit card. Microsoft credits ₹14,500 INR or 200 USD for the first month in the azure portal and the popular services are free for one year. For this free service activation, you need to register your Azure account using your credit/debit card free of cost.

We can also activate Azure subscription through Microsoft Visual Studio. If you have Visual Studio other than community edition then go to Visual Studio website (https://my.visualstudio.com/) and click on the **Benefits** menu and activate **Azure free $50 monthly credit** in your Azure account. This will be helpful for most of the paid service access in the Azure portal to develop an application for testing purposes because we are using very limited credit and once the free access is over, then all services will stop working. Remember that every Azure service has been chargeable and the amount will be debited from the monthly credited amount in azure portal. So, once it's over then we won't be able to access any of the paid services in the particular month and need to wait for the next month for the new credit.

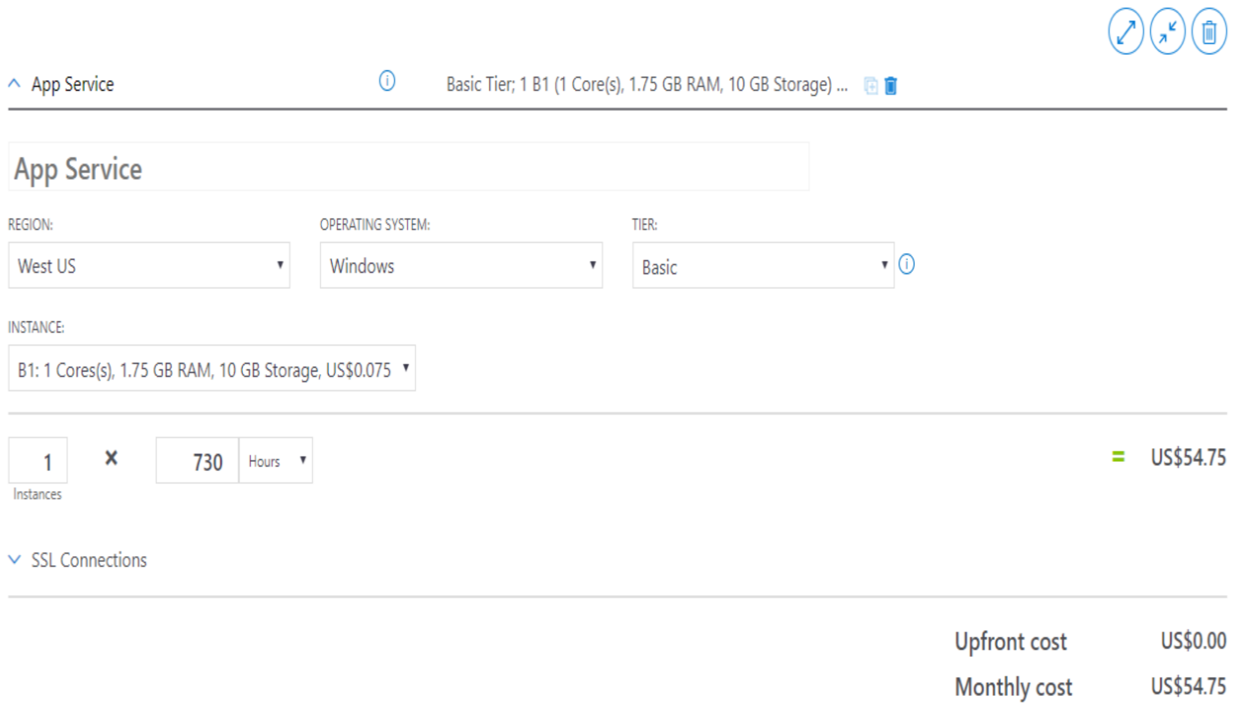
Important Tips: We can delete unwanted services from the Azure portal to avoid and save the Azure free credited cost. In this way, we can utilize the Azure free credited amount for every month. Also try to disable application insight or any other unwanted options while creating an app service, resources, virtual machine, etc., in the Azure portal.

Azure Pricing Calculator

The pricing tier will be considered as a major concern when you become an Azure developer because everything will be chargeable and nothing will be free in the current infrastructure for commercial use. One of the main preferences of a customer is cost reductions for the infrastructure setup and we need to take care of that in a proper and efficient manner.

The Azure pricing calculator will help you to estimate the actual cost charges in each and every service in the Azure portal in days, hours, and months.

- https://azure.microsoft.com/en-in/pricing/calculator/


App Service — Basic Tier; 1 B1 (1 Core(s), 1.75 GB RAM, 10 GB Storage) ...

App Service

REGION: West US

OPERATING SYSTEM: Windows

TIER: Basic

INSTANCE: B1: 1 Cores(s), 1.75 GB RAM, 10 GB Storage, US$0.075

1 Instances × 730 Hours = US$54.75

SSL Connections

Upfront cost	US$0.00
Monthly cost	US$54.75

Note: Select the region based on the customer need or nearest to customer location; this will reduce the latency.

Cognitive Services

Cognitive Services are many sets of machine learning algorithms developed by Microsoft for solving problems in the Artificial Intelligence (AI) field. Microsoft is providing various cognitive services as a package through an API or an SDK in the real world. This will be more helpful for an end-user who doesn't have expertise in machine learning, because anyone can simply use Cognitive Services API using a rest client call.

Cognitive Services APIs are grouped into five categories.

- Decision
- Vision
- Speech
- Language
- Search

Note: Cognitive Services will be available at least 99.9% of the time and No SLA is provided for the free tier.

What is LUIS?

One of the prebuilt Artificial Intelligence Cognitive Services is called Language Understanding Intelligent Service, or LUIS, and this is a part of the cognitive services language category. LUIS will create a strong conversation layer in the application and is a very intelligent way of communication. As we know, LUIS is a Spanish name, and it is a natural language processing service that enables us to understand human language such as in our own application, website, chatbot, IoT device, etc. Once you configure, train and publish your LUIS model, then the application can easily receive user input in natural language and take an appropriate action based on the intent, utterances, and entity configured in the LUIS Model. This type of innovation will be helpful for lots of new enhancement development in the future.

We can easily create LUIS models with the help of a https://www.luis.ai/ account and for this we require LUIS "Authoring resource" in Azure. So first of all, we need to create a LUIS API in Azure account using a valid subscription key.

Why LUIS?

The reason for using LUIS in the application Instead of choosing other platforms is because it provides more features for artificial intelligence than other platforms. The following are the important benefits for using LUIS in the application.

- Highly secure and hosted in Azure.
- Easy to scale (Up & Out) and hosted in Azure.
- LUIS provides a simple user interface and is very easy to use.
- We can build LUIS applications with fewer lines of codes.
- Strong conversational layer in the application.

- Continuously improve natural language models.
- We can give more accuracy for the utterances.
- Build an intelligent application with less time.
- AI expertise is not required for handling the LUIS application because all are prebuilt AI.
- Easy to retrain and build an intelligent application.
- More machine learning features are available like ML entities, etc.

What's new for Language Understanding (LUIS)?

As per the latest Microsoft build 2020, Microsoft has announced new features for Language Understanding (LUIS) in the application.

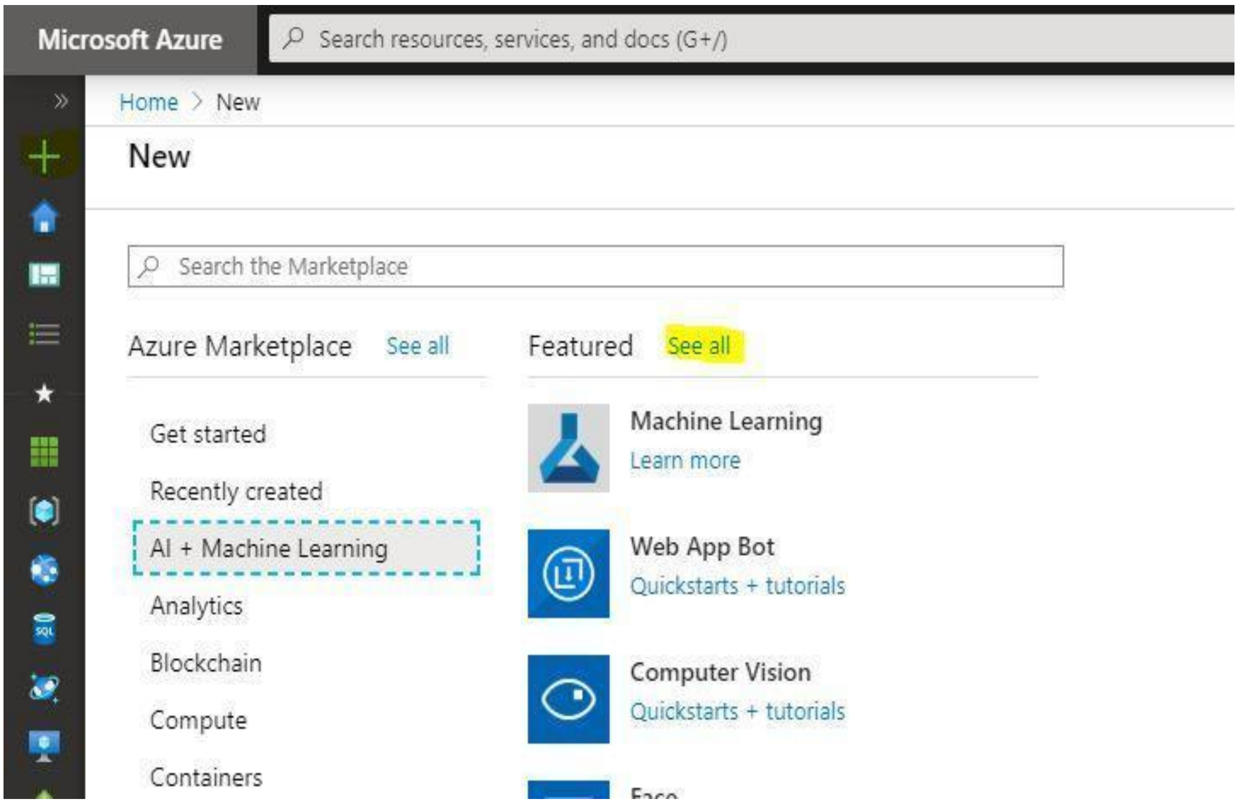
We can check the following features of LUIS announced by Microsoft.

- The user labelling experience improved in the new version of LUIS.
- The new style of graphical representation added in entity mapping in LUIS.
- There is a seamless upgrade for existing LUIS applications.
- The more powerful ML entities with hierarchy instead of composite entities.
- Add entity as a ML feature list in LUIS application and map through the intent level.

How to create a LUIS Authoring Service?

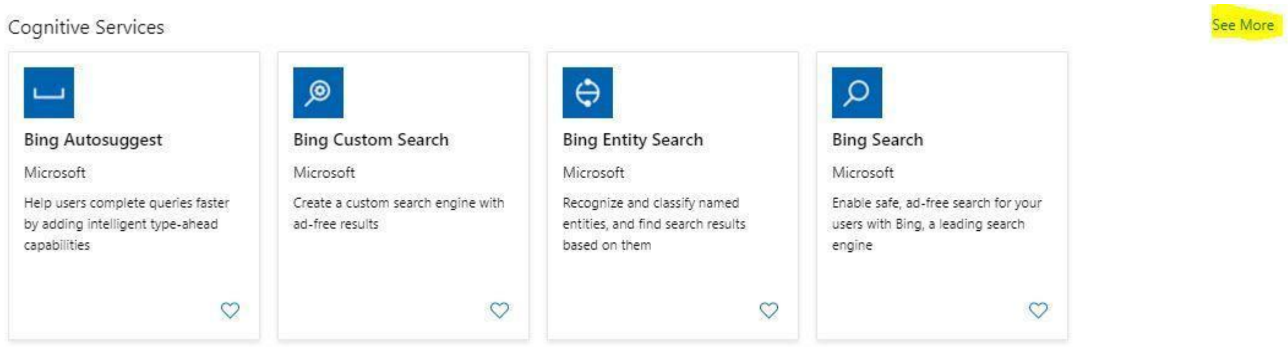
Step 1:

Go to Azure portal and click on **"+"** icon -> go to **"AI + Machine Learning"** -> Click on **"See all"**.



Step 2:

If you are not able to see LUIS in the "Cognitive Services" then click on the "See more" link on the right side.


Cognitive Services

See More

Bing Autosuggest
Microsoft
Help users complete queries faster by adding intelligent type-ahead capabilities

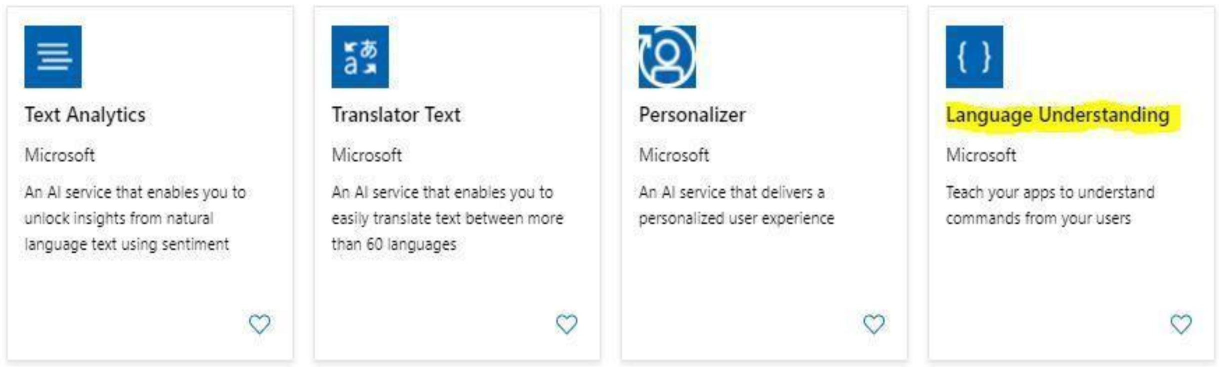
Bing Custom Search
Microsoft
Create a custom search engine with ad-free results

Bing Entity Search
Microsoft
Recognize and classify named entities, and find search results based on them

Bing Search
Microsoft
Enable safe, ad-free search for your users with Bing, a leading search engine

Step 3:

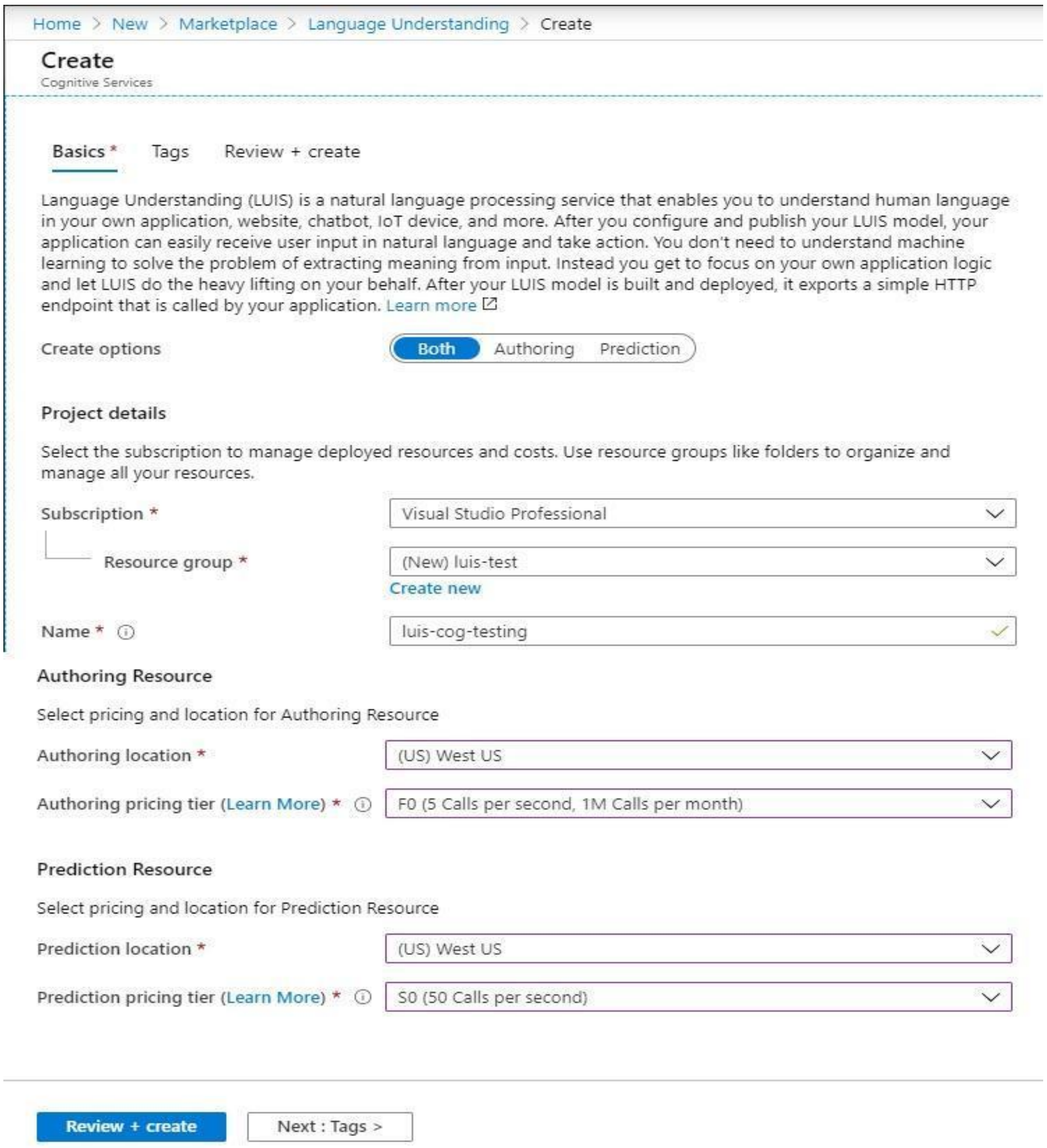
Once you are able to see Language Understanding in the "Cognitive Services" list then go and click on it.


Text Analytics
Microsoft
An AI service that enables you to unlock insights from natural language text using sentiment
Translator Text
Microsoft
An AI service that enables you to easily translate text between more than 60 languages
Personalizer
Microsoft
An AI service that delivers a personalized user experience
Language Understanding
Microsoft
Teach your apps to understand commands from your users

Step 4:

The following screen will appear once you click on the "Create" button in "Language Understanding"

- **Subscription:** We can select our own Azure subscription for Language Understanding and if you don't have the subscription then create a new one, either free or paid.
- **Resource group:** Group together all the LUIS related resources into one place and this will be helpful for filtering and applying the permission level (RBAC, LOCK) for the entire group in the LUIS application. So, we can create a new resource group or choose from an existing one (We selected our existing resource group as "luis-test").
- **Authoring location:** We can select our location of Authoring location. The best thing is we can choose a location closest to our customer needs.
- **Authoring pricing tier:** We can choose the appropriate pricing tier as per our needs.
- **Prediction location:** We can select our location of prediction. The best thing is we can choose a location closest to our customer needs.
- **Prediction pricing tier:** As of now there are two pricing tiers available, "FO" & "SO" and obviously "FO" is the free one and we can choose the appropriate pricing tier as per our needs. When you are planning to do a performance testing then choose “SO” as the priority one because it will handle a greater number of hits per second otherwise choose “FO” more learning purposes.


Home > New > Marketplace > Language Understanding > Create

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 *: Visual Studio Professional
Resource group *: (New) luis-test
Create new
Name *: luis-cog-testing

Authoring Resource

Select pricing and location for Authoring Resource

Authoring location *: (US) West US
Authoring pricing tier (Learn More) *: F0 (5 Calls per second, 1M Calls per month)

Prediction Resource

Select pricing and location for Prediction Resource

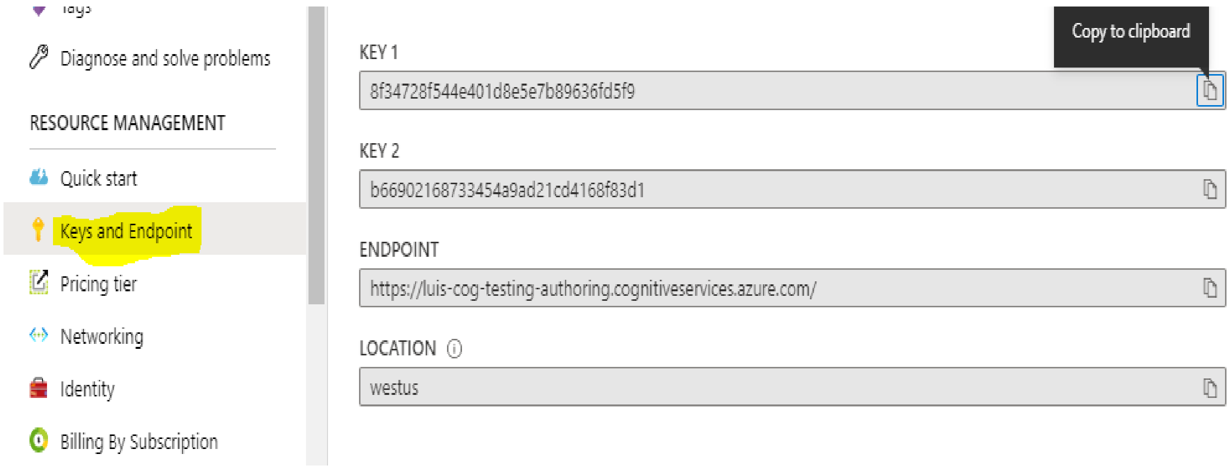
Prediction location *: (US) West US
Prediction pricing tier (Learn More) *: S0 (50 Calls per second)

- Click on the "Review + Create" button and wait for the build success.
- Once the build has succeeded, then click on the "Dashboard" and we can see "luis-cog-testing" is created in the All resources list. LUIS is ready for use!

The uses of LUIS Endpoint Key

The endpoint key will be generated while creation of Authoring resources in LUIS Cognitive services in Azure portal. These keys are used to access your Cognitive Service API for doing all types of CRUD operations. Do not share your keys with the public, rather store them securely using Azure Key Vault or any secure way.

The “Keys and Endpoint” menu is part of the Resource Management section in the LUIS Cognitive service in Azure portal. We can consider any of the keys for accessing the Cognitive Service API and also regenerate the key as per your needs.



KEY 1
8f34728f544e401d8e5e7b89636fd5f9

KEY 2
b66902168733454a9ad21cd4168f83d1

ENDPOINT
https://luis-cog-testing-authoring.cognitiveservices.azure.com/

LOCATION
westus

Create Luis Application

We have already created the LUIS authoring resource in Azure and now we can easily create LUIS model in Luis account. So, go to your LUIS account and create a new LUIS App.

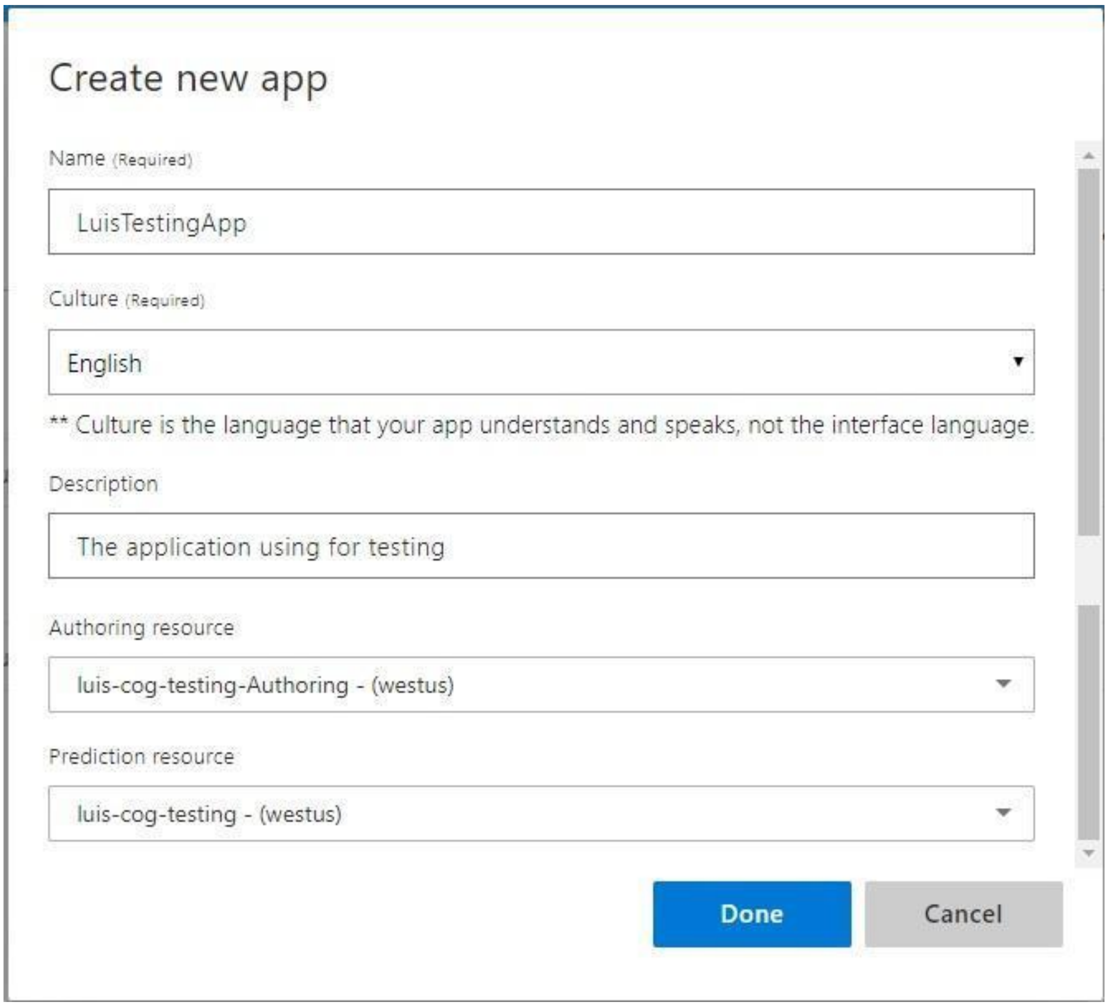
Name: Name of the Luis application.

Culture: The current culture or language you’re going to use in Luis application.

Description: A short description of our application.

Authoring resource: The resource that we have created in Azure.

Prediction resource: The resource that we have created in Azure.


Create new app

Name (Required)

LuisTestingApp

Culture (Required)

English

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

Description

The application using for testing

Authoring resource

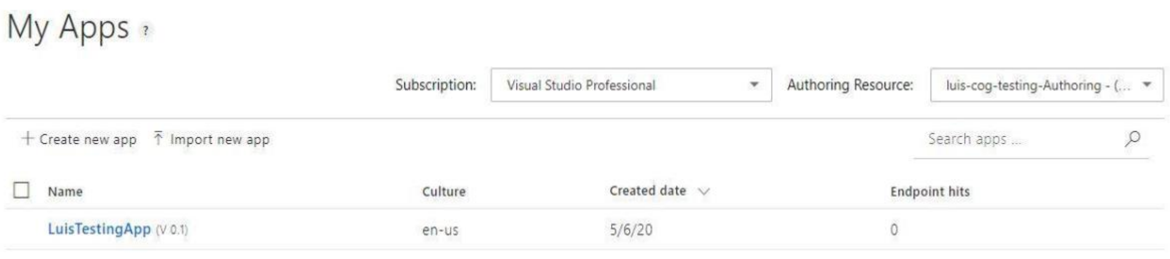
luis-cog-testing-Authoring - (westus)

Prediction resource

luis-cog-testing - (westus)

Output:

The app is successfully created in LUIS and by default it will contain one "Intent" called "None". We will have a detailed discussion over the topics: intent, utterances, entity.


My Apps

Subscription: Visual Studio Professional

Authoring Resource: luis-cog-testing-Authoring - (...

Name	Culture	Created date	Endpoint hits
LuisTestingApp (V 0.1)	en-us	5/6/20	0

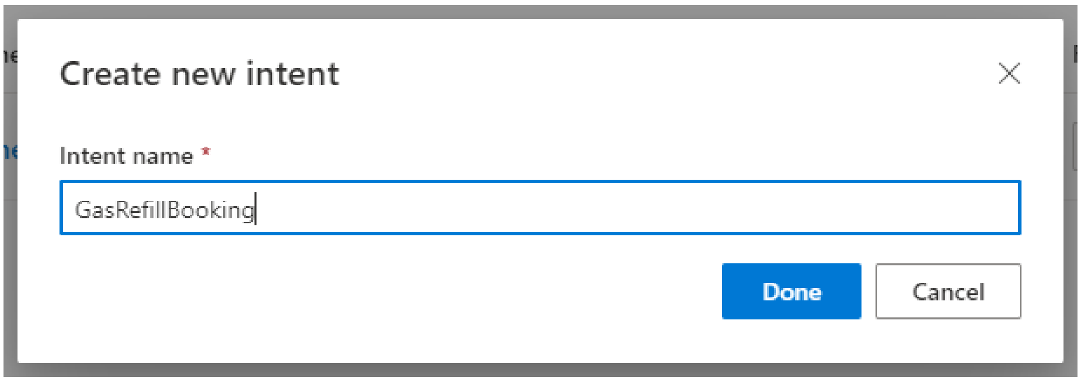
What is Intent?

Intent is represented as a task or the user wants to perform an action like OrderingFood , OrderingGrocery, GasRefillBooking ,etc.

In the real world, we have used more numbers of intents in the Covid-19 pandemic situation because during the lockdown period we have used many online services for purchase, booking, bill payments, etc. All these activities are internally performing many kinds of tasks. These tasks are created as an intent in the LUIS application for various purposes.

How to create an intent?

Step 1: Click on the LUIS application.
Step 2: Go and click on "+ Create".
Step 3: We have given the intent name as "GasRefillBooking".
Step 4: Click on "Done".


Create new intent

Intent name *

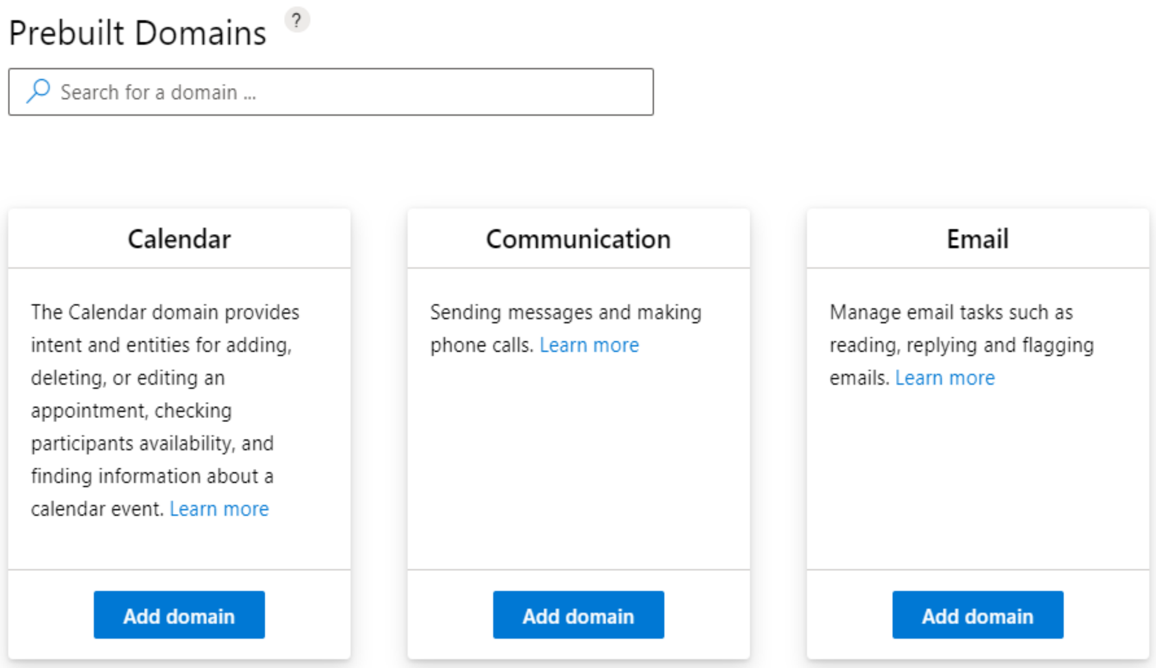
GasRefillBooking

Prebuilt domains in LUIS application?

The prebuilt domain already has a set of intents and utterances for the respective functional activities like each task has appropriate actions. The following are the prebuilt domains available in the LUIS applications and all are used for our real-time activities.

- Calendar
- Communication
- Email
- HomeAutomation
- Note
- Places
- RestaurantReservation
- ToDo
- Utilities
- Weather
- Web

We can add prebuilt domains by clicking on “Add domain” button and once added the entity will be included in the respective LUIS application.



The Calendar domain provides intent and entities for adding, deleting, or editing an appointment, checking participants availability, and finding information about a calendar event. Learn more

Sending messages and making phone calls. Learn more

Manage email tasks such as reading, replying and flagging emails. Learn more

Note: We can easily remove the prebuilt domain entity from the LUIS application by clicking on the “Remove domain” button which appears on the already added prebuilt domain entity.

E.g. The following JSON response is the utterance of “What is the current weather in Bangalore” in the LUIS application that we have created and also this utterance matching top intent is part of the already-added prebuilt domain.

```
{
  "query": "\"What is the current weather in Bangalore\"",
  "prediction": {
    "topIntent": "Weather.CheckWeatherValue",
    "intents": {
      "Weather.CheckWeatherValue": {
        "score": 0.406864345
      },
      "Weather.GetWeatherAdvisory": {
        "score": 0.06625718
      },
      "Weather.CheckWeatherTime": {
        "score": 0.0321165435
      },
      "Weather.QueryWeather": {
        "score": 0.009715379
      },        "Weather.ChangeTemperatureUnit": {
        "score": 0.008676363
      },
      "None": {
        "score": 0.00400842959
      },      },
    "entities": {
      "geographyV2": [
        {
          "value": "Bangalore",
          "type": "city"
        }
      ],
      "$instance": {
        "geographyV2": [
          {
            "type": "builtin.geographyV2.city",
            "text": "Bangalore",
            "startIndex": 32,
            "length": 9,
            "modelTypeId": 2,
            "modelType": "Prebuilt Entity Extractor",
            "recognitionSources": [
              "model"
            ]
          }
        ]
      }
    }
  }
}
```

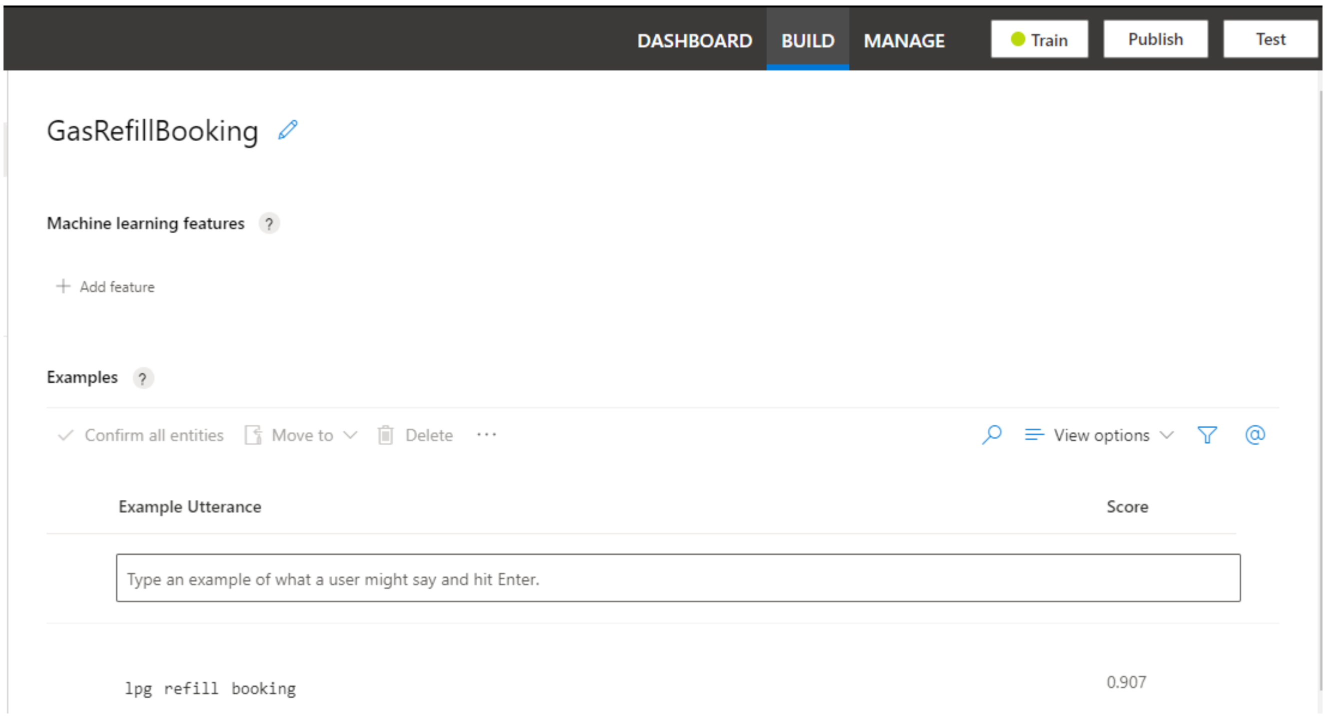
What are Utterances?

All the intent requires some kind of explanation, otherwise it will be meaningless so utterances are the explanation or the detailed description of the intent. LUIS will contain unique utterances across all the intents. When you try to add duplicate utterances in another intent or the same intent then LUIS will automatically remove the duplicate utterance from the application. This will improve the consistency and intelligence of the NLP engine and give us more accuracy in the LUIS application.

Note: “To train the LUIS application at least one utterance is required”

How to create an Utterance?

Step 1: Click on the LUIS application.
Step 2: Click on the intent. E.g. GasRefillBooking
Step 3: Add the utterances as “lpg refill booking”.
Step 4: Press on Keyboard “Enter” key.
Step 5: The utterances are created successfully.



What are Entities?

The entities are variables or parameters in the LUIS application which will pass important information from the utterances that we have given. There are many ways we can create entities for different activities in the LUIS application which is done to improve the machine learning capabilities.

Types of entity creation

The following are the various types of Entity creation in LUIS.

- Create a new entity (Using "Create" button).
- Add a prebuilt entity.
- Add prebuilt domain entity.

Create new entity

There are a few kinds of entity creation in the "Create new entity" option and the following are the different types of entity creation.

- Machine learned entities
- List
- Regex
- Pattern.Any

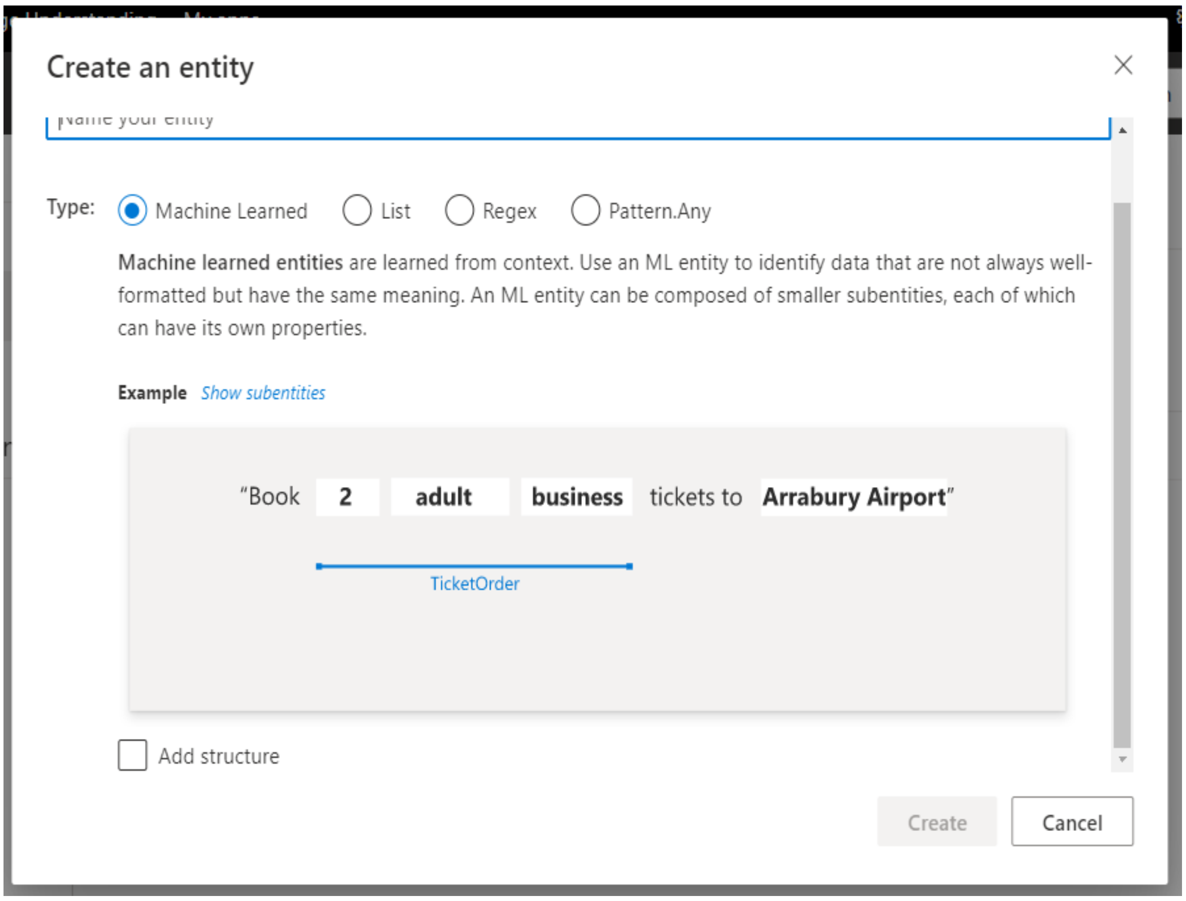
Note: Microsoft removed "Simple, Hierarchical, Composite" entities from the entity creation list in MS Build 2020. And a list of improvised machine learning entities was announced for the same.

Machine Learned entities

Machine Learned entities are the new way of entity creation updated by Microsoft on MS Build 2020:

"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 sub entities, each of which can have its own properties."

Without clicking on “Add structure” checkbox, create an entity in the Machine Learned type, then it will create a simple entity similar to the older version of LUIS.


Create an entity

Name your entity

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 *Show subentities*

“Book **2** **adult** **business** tickets to **Arrabury Airport**”

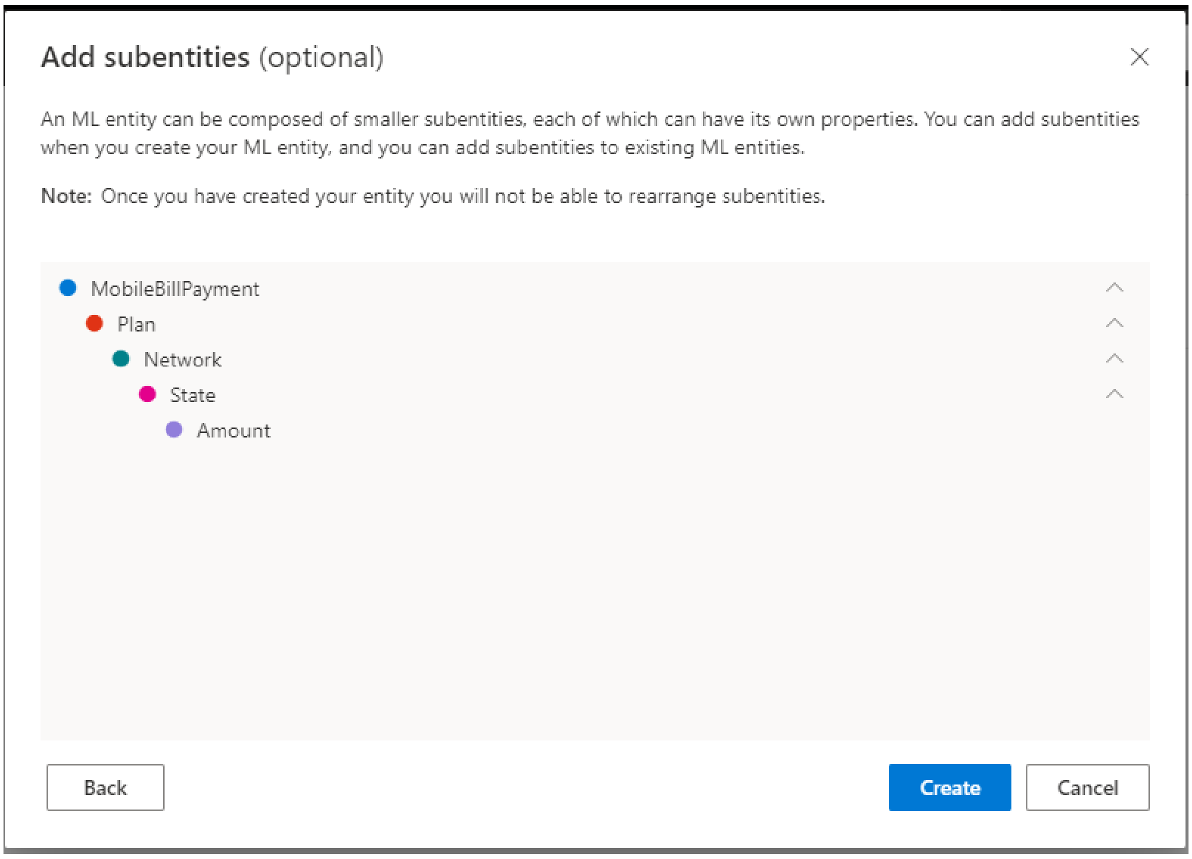
☐ Add structure

For example, if the user's intent is “GasRefillBooking” and the utterance is “I need to refill my lpg 4567” then you can map “4567” as bookingId and it will be recognized as an entity. This will capture booking details for the customer.

Subentities in Machine Learned Entities

While creating a Machine learned entity, there is an option called “Add structure” checkbox. When you click on “Add structure” checkbox then it will create a sub entities option for the given Machine learned entity and this will be similar to the older version of LUIS application entities like Composite and Hierarchical.

The name itself describes it, as it is a part by part information into the object, using a Machine learned entity to represent an object that has many parts. The ML entity is made up of entities that form the whole object information. For example, a ML entity called “MobileBillPayment” in an application and it can be composed of four child entities that describe attributes of the bill to payment: Plan, Network, State and Amount.



Add subentities (optional)

An ML entity can be composed of smaller subentities, each of which can have its own properties. You can add subentities when you create your ML entity, and you can add subentities to existing ML entities.

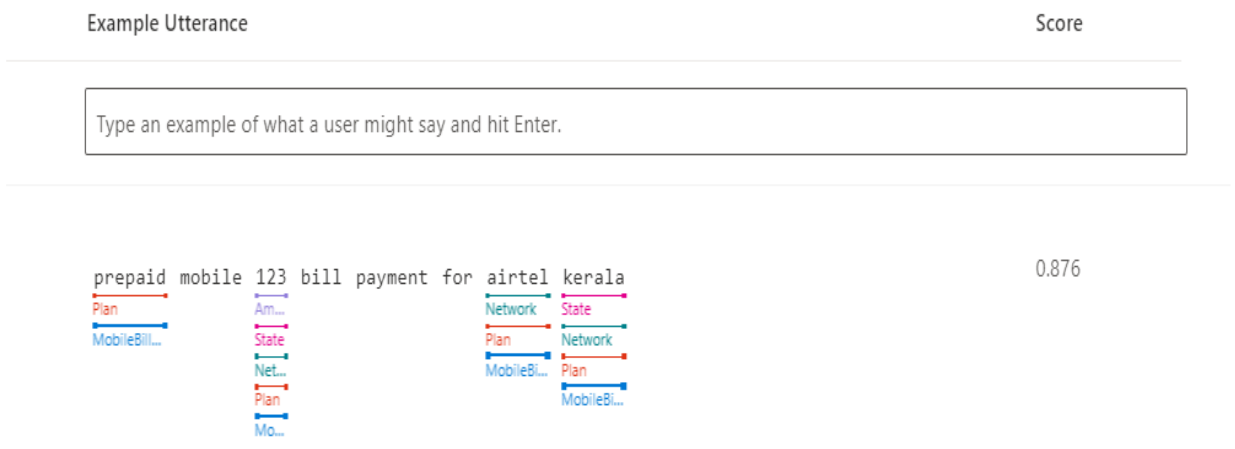
Note: Once you have created your entity you will not be able to rearrange subentities.

- MobileBillPayment
  - Plan
    - Network
      - State
        - Amount

Note: Once you have created your entity you will not be able to rearrange subentities.

Subentities mapping in Utterances

The created Machine-Learned sub entities are manually mapped into the given utterance “prepaid mobile 123 bill payment for airtel kerala”. The new graphical representation is clearly displayed as the ML entity details and will give us all the information about the utterance.



Example Utterance

Score

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

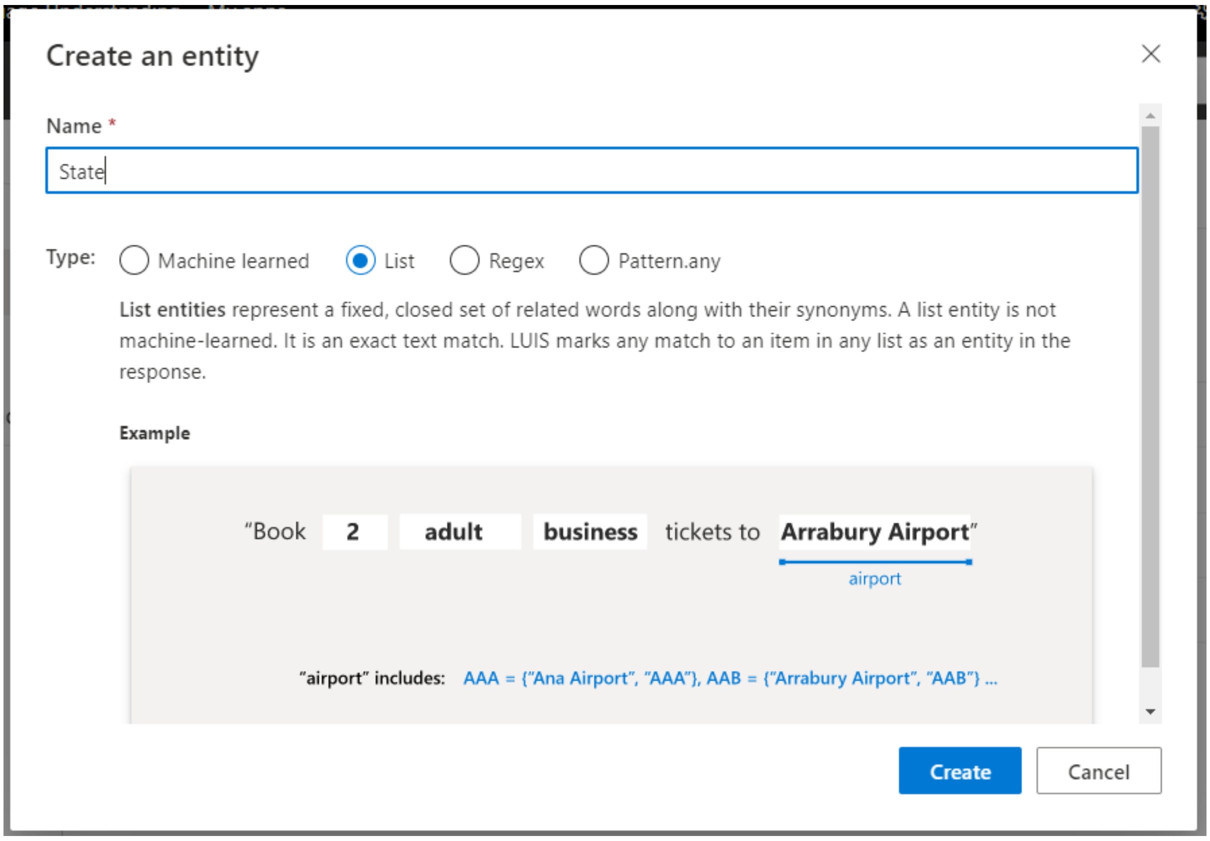
prepaid mobile 123 bill payment for airtel kerala

0.876

LIST

List entities are represented as a fixed entity and it describes the set of related words mapped together or along with their synonyms. It clearly says that synonymous is always representing the same meaningful words and it is interchangeable. The list entity always returns the response for the exact text match because a list entity isn't part of a machine-learned entity.

Click on the Entities button in the menu list for the respective LUIS application and it will open a popup window with type of entity information. Select the List entity radio button and Click on “Create” button.


Create an entity

Name *

State

Type: Machine learned [✓ List] Regex Pattern.any

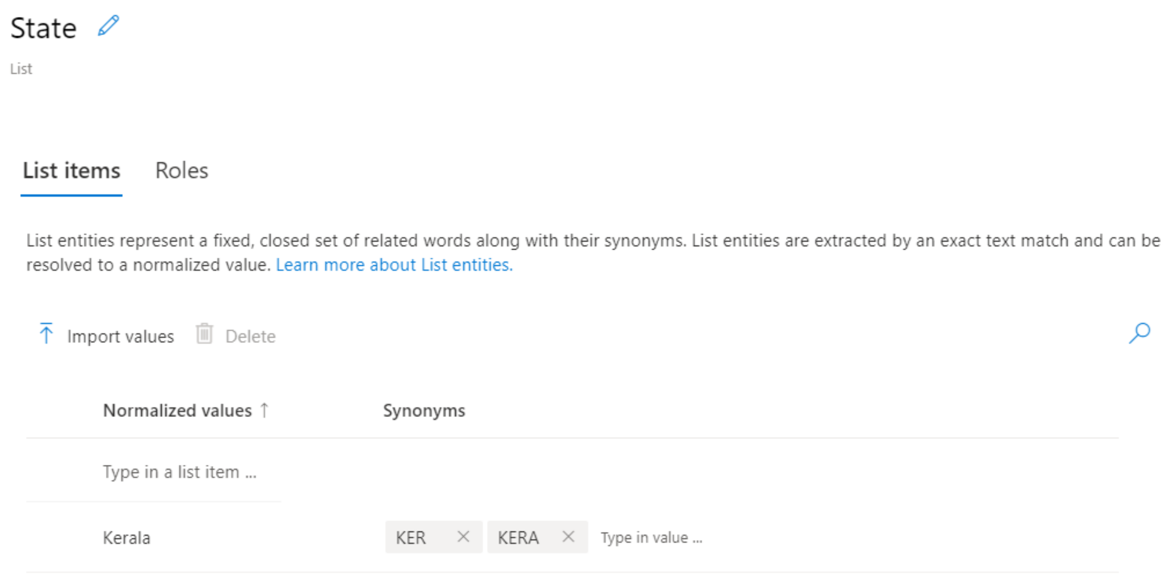
List entities represent a fixed, closed set of related words along with their synonyms. A list entity is not machine-learned. It is an exact text match. LUIS marks any match to an item in any list as an entity in the response.

Example

"Book **2** **adult** **business** tickets to **Arrabury Airport**"

"airport" includes: AAA = {"Ana Airport", "AAA"}, AAB = {"Arrabury Airport", "AAB"} ...

Once created It will display the add list item option in the already created "State" LIST entity. So, we can add any number of normalized values in the LIST entity section and we have added "Kerala" as the normalized value and Synonyms as "KER", "KER", etc. All the list items are part of the parent entity and there's no need to map or label LIST entities to utterances because this will create a response based on the exact text match.


State

List

List entities represent a fixed, closed set of related words along with their synonyms. List entities are extracted by an exact text match and can be resolved to a normalized value. Learn more about List entities.

Normalized values ↑	Synonyms
Type in a list item ...	
Kerala	KER, KERA, Type in value ...

Here is the JSON response for LIST entity and it will automatically recognize the LIST entity information for the given utterance “Where is ker”.

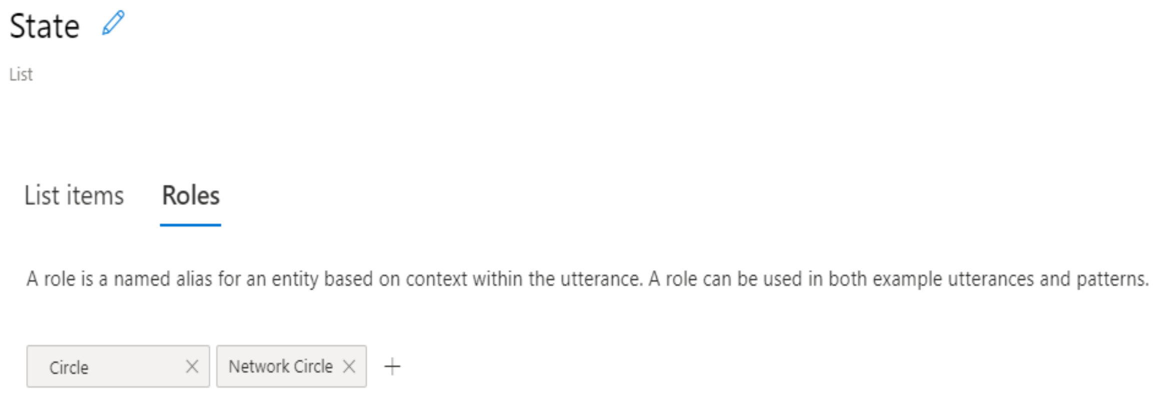
```
{
  "query": "\"Where is ker\"",
  "prediction": {
    "topIntent": "None",
    "intents": {
      "None": {
        "score": 0.9559727
      },
      "BillPayment": {
        "score": 0.0251173135
      },
      "GasRefillBooking": {
        "score": 0.0124031492
      }
    },
    "entities": {
      "State": [
        [
          "Kerala"
        ]
      ],
      "$instance": {
        "State": [
          {
            "type": "State",
            "text": "ker",
            "startIndex": 10,
            "length": 3,
            "modelTypeId": 5,
            "modelType": "List Entity Extractor",
            "recognitionSources": [
              "model"
            ]
          }
        ]
      }
    }
  }
}
```

Add a role for an entity

The role is a named subtype of an entity, based on context.

We have added “Circle” and “Network Circle” as the Role or named subtype of State List entity and this will easily recognize Kerala as the network circle for the respective activities.

E.g. For mobile prepaid or postpaid recharge we need to select the network circle.


State

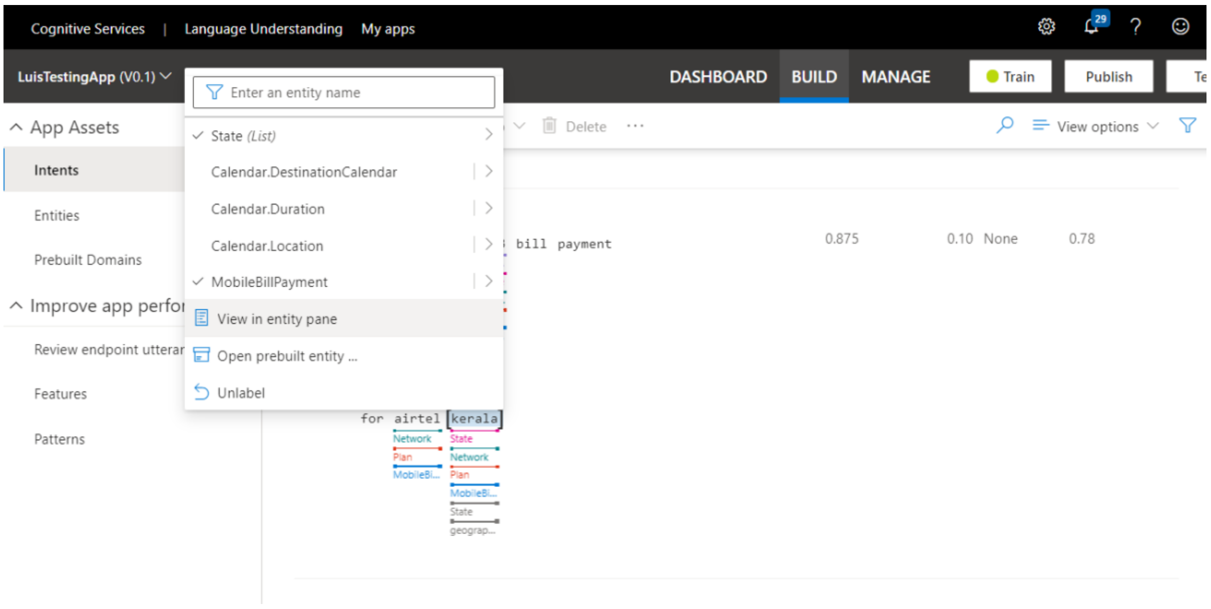
List

List items Roles

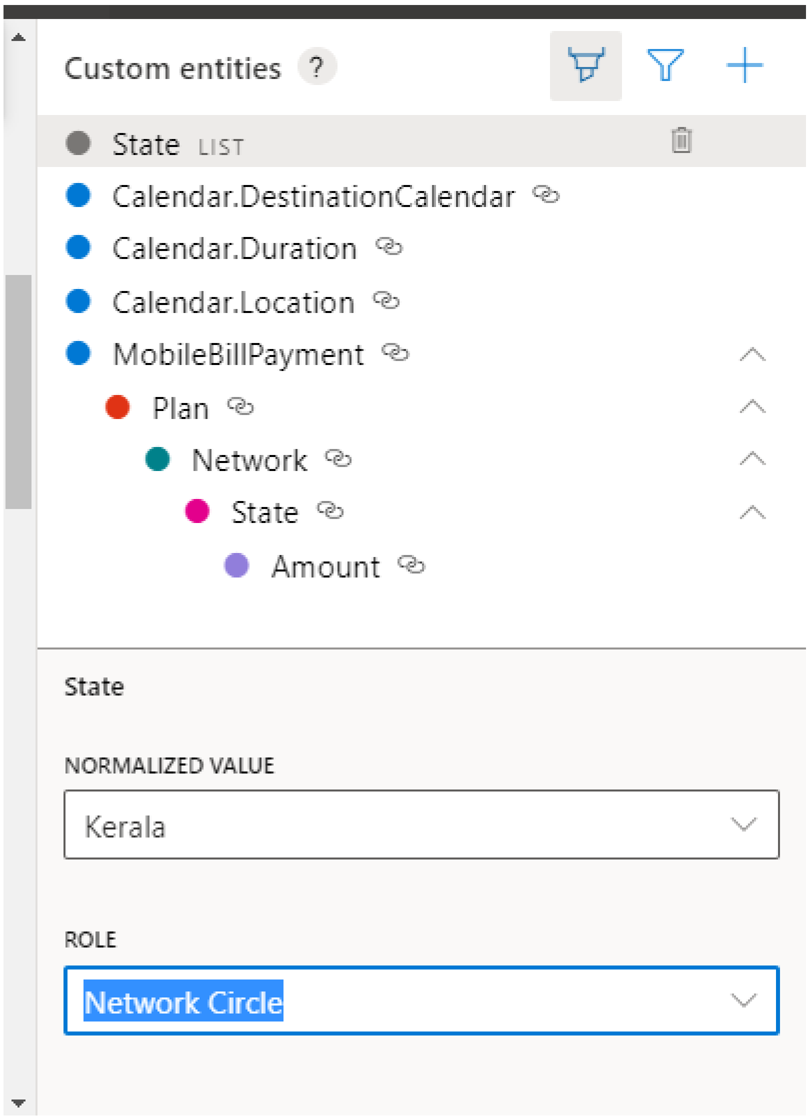
A role is a named alias for an entity based on context within the utterance. A role can be used in both example utterances and patterns.

Circle ✕ Network Circle ✕ +

After successful building and training, go to the utterances for mapping the "Role". In the following example we have clicked on the "kerala" and It will open a palette. Click on the "View in entity pane" list menu and it will open a tree structure for all the entities in the LUIS application.



The "View in entity pane" will open the entity tree structure and click on the State List Entity and then go to the bottom of the palette and select the role and it will display the synonyms of the respective role added in the State List entity.



Custom entities

State LIST

Calendar.DestinationCalendar

Calendar.Duration

Calendar.Location

MobileBillPayment

Plan

Network

State

Amount

State

NORMALIZED VALUE

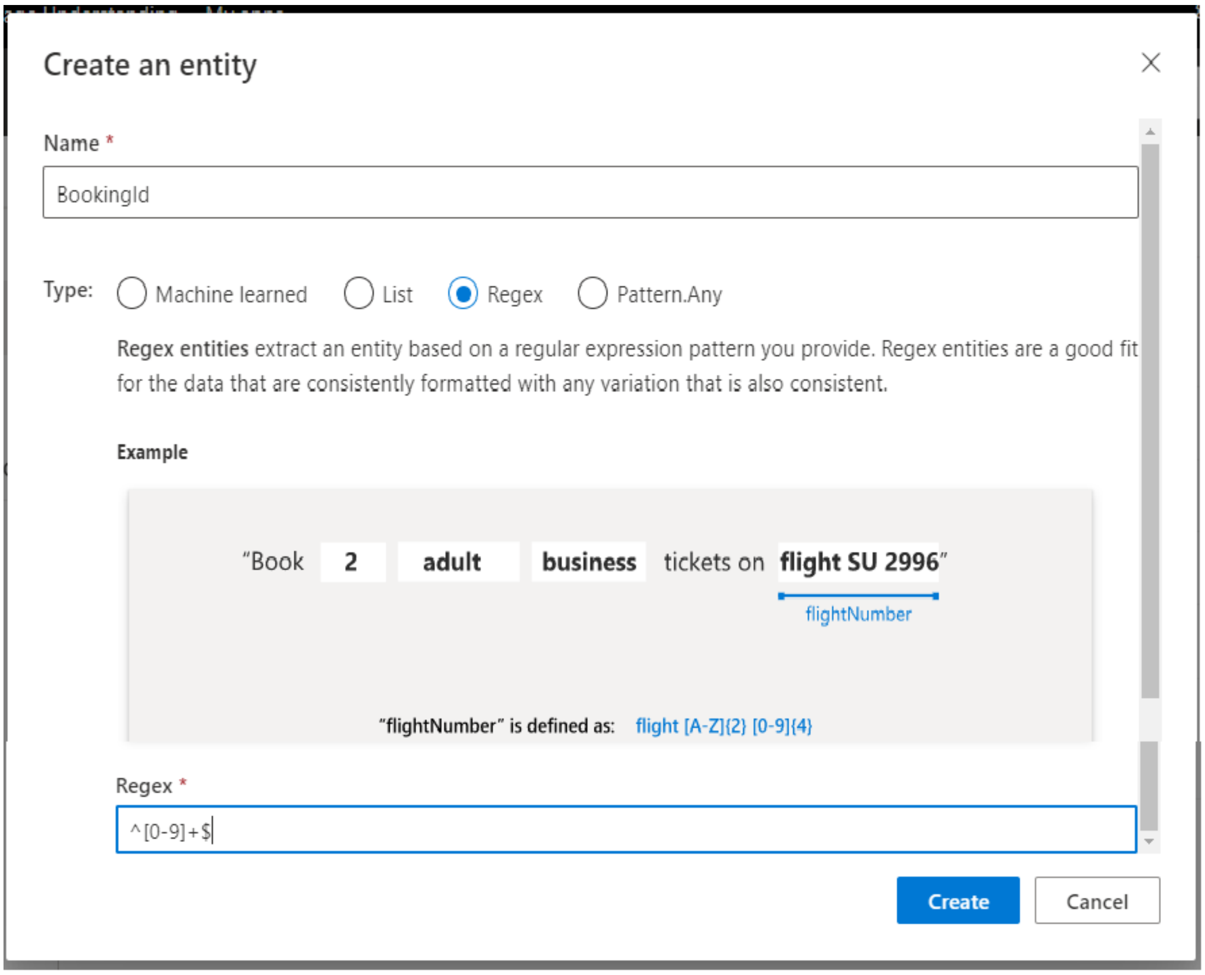
Kerala

ROLE

Network Circle

Regex

Regex entities extract an entity based on a regular expression pattern. That is to say, we are extracting a specific entity or a parameter from the utterances for a specific action execution. This will give more consistency and accuracy for the extraction of an entity from the utterances.


Create an entity

Name *

BookingId

Type: ◯ Machine learned ◯ List [✓ Regex] ◯ Pattern.Any

Regex entities extract an entity based on a regular expression pattern you provide. Regex entities are a good fit for the data that are consistently formatted with any variation that is also consistent.

Example

"Book **2** **adult** **business** tickets on **flight SU 2996**"

flightNumber

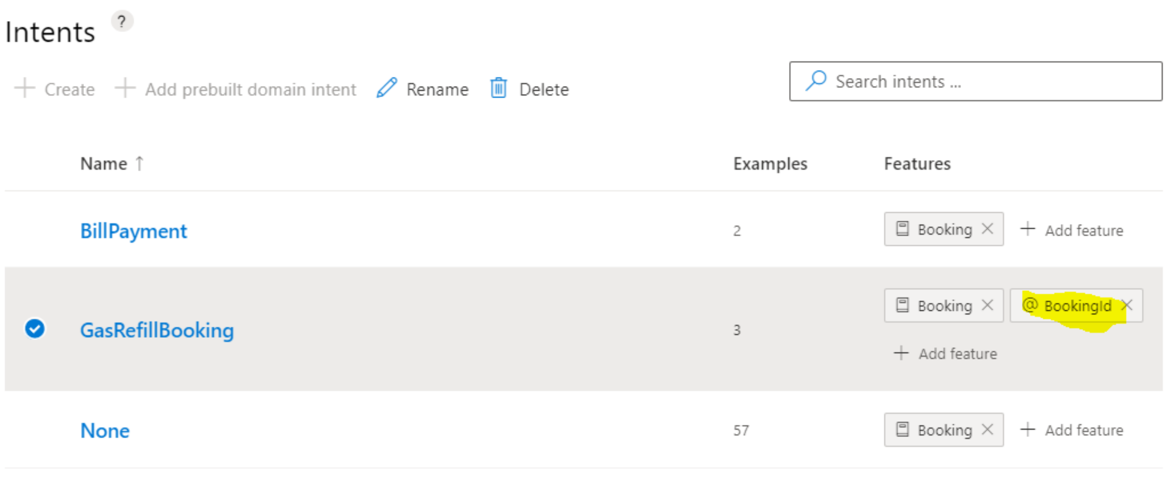
"flightNumber" is defined as: flight [A-Z]{2} [0-9]{4}

Regex *

^[0-9]+$

E.g. The Regex "^[0-9]+$" only accepts numbers.

The "BookingId" we can add as a global feature in any of the entities in the LUIS application. This will be similar to what we have added to the phrase list in the LUIS app. Click on "Add feature" and add a phrase list or already-created entity in the respective intent.


Intents

Create | Add prebuilt domain intent | Rename | Delete | Search intents ...

Name	Examples	Features
BillPayment	2	Booking; Add feature
GasRefillBooking	3	Booking; BookingId; Add feature
None	57	Booking; Add feature

Here is the json response of “What is the booking id 25678” utterances and it will return accurate entity information with the help of Regex that we have added.

```
{
  "query": "\"What is the booking id 25678\"",
  "prediction": {
    "topIntent": "Weather.GetWeatherAdvisory",
    "intents": {
      "Weather.GetWeatherAdvisory": {
        "score": 0.0530414023
      },
      "Weather.CheckWeatherTime": {
        "score": 0.0437874235
      },
      "Weather.CheckWeatherValue": {
        "score": 0.036190737
      },
      "Weather.QueryWeather": {
        "score": 0.0283024088
      },
      "GasRefillBooking": {
        "score": 0.0206228029
      },
      "None": {
        "score": 0.0112512978
      },
      "BillPayment": {
        "score": 0.0102754943
      },
      "Weather.ChangeTemperatureUnit": {
        "score": 0.004960226
      }
    },
    "entities": {
      "dimension": [
        {
          "number": 25678,
          "units": "Inch"
        }
      ],
      "$instance": {
        "dimension": [
          {
            "type": "builtin.dimension",
            "text": "25678\"",
            "startIndex": 24,
                "length": 6,
                "modelTypeId": 2,
                "modelType": "Prebuilt Entity Extractor",
                "recognitionSources": [
                  "model"
                ]
              }
            ]
          }
        }
      }
    }
```

Here is the json response of “What is the booking id “X25678” utterance and it will never return accurate entity information because we have added the regex pattern which will only accept numbers.

```
{
  "query": "\"What is the booking id X25678\"",
  "prediction": {
    "topIntent": "Weather.GetWeatherAdvisory",
    "intents": {
      "Weather.GetWeatherAdvisory": {
        "score": 0.0530414023
      },
      "Weather.CheckWeatherTime": {
        "score": 0.0437874235
      },
      "Weather.CheckWeatherValue": {
        "score": 0.036190737
```

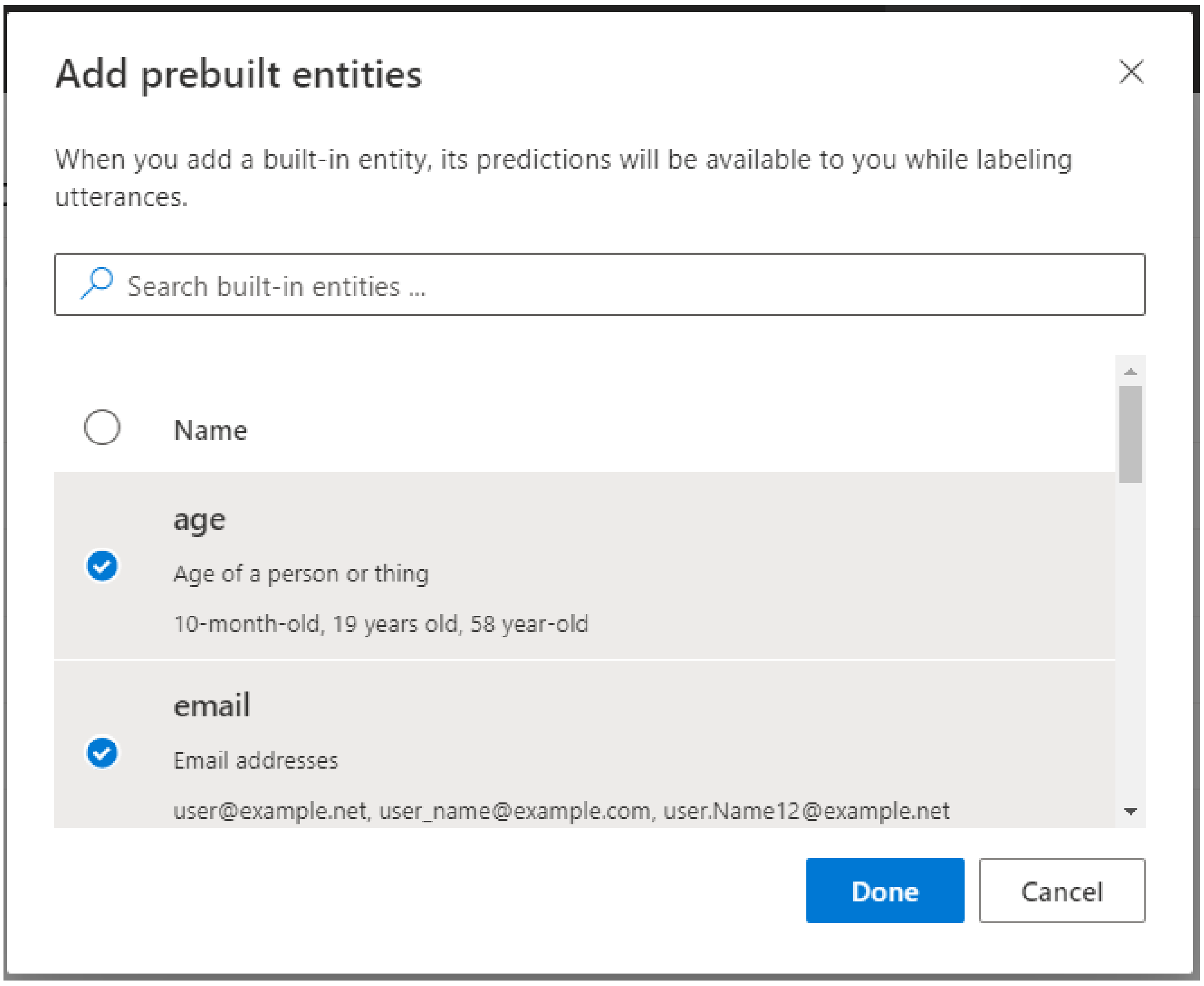
```
            },
            "Weather.QueryWeather": {
              "score": 0.0283024088
            },
            "GasRefillBooking": {
              "score": 0.0206228029
            },
            "None": {
              "score": 0.0112512978
            },
            "BillPayment": {
              "score": 0.0102754943
            },
            "Weather.ChangeTemperatureUnit": {
              "score": 0.004960226
            }
          },
          "entities": {}
        }
      }
```

Add prebuilt entity

The prebuilt entity is when you add a built-in entity, its predictions will be available to the entity mapped or labelled utterances. The following are the currently available prebuilt entities in the LUIS application.

- age
- email
- keyPhrase

- money
- number
- ordinal
- ordinalV2
- percentage
- phonenumber
- url


Add prebuilt entities

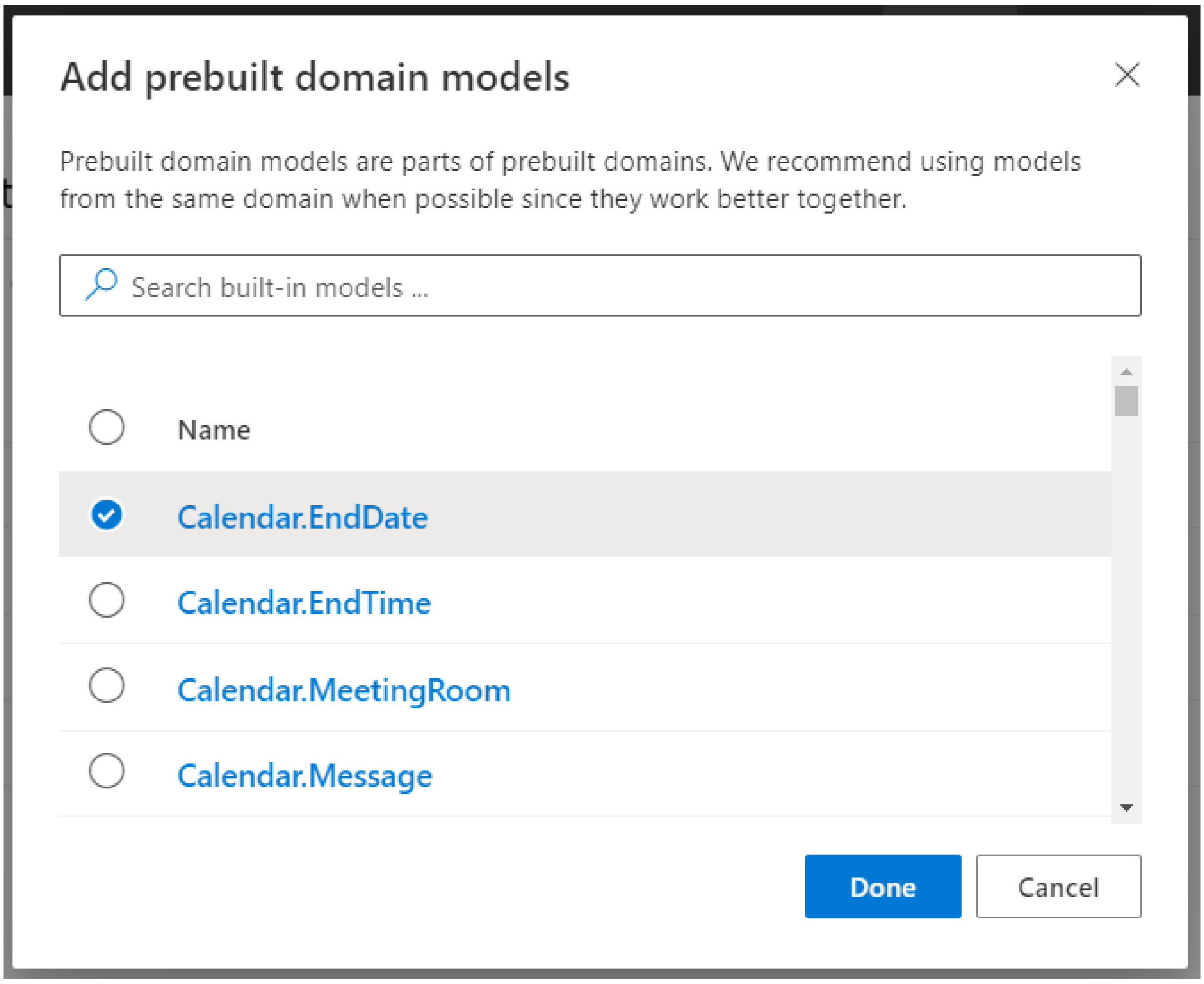
When you add a built-in entity, its predictions will be available to you while labeling utterances.

Search built-in entities ...

	Name
✓	age Age of a person or thing 10-month-old, 19 years old, 58 year-old
✓	email Email addresses user@example.net, user_name@example.com, user.Name12@example.net

Add prebuilt domain entity

The prebuilt domain entity already has a set of intents and utterances, etc for the respective functional activities. The prebuilt domain models are part of prebuilt domains. It is recommended to use models from the same domain when possible since they work better together.


Add prebuilt domain models

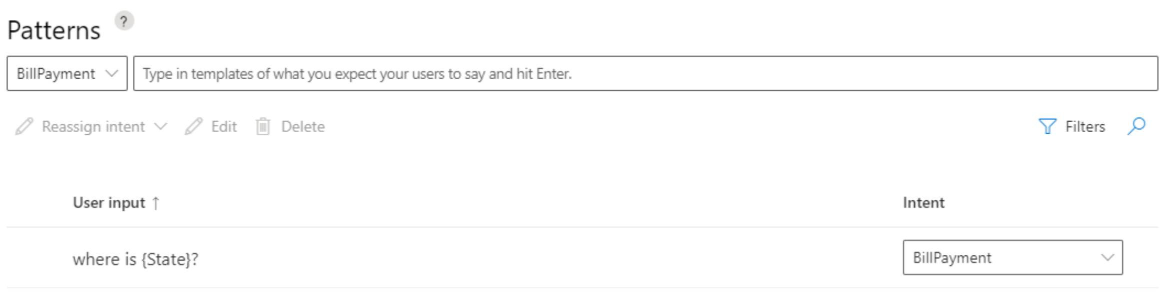
Prebuilt domain models are parts of prebuilt domains. We recommend using models from the same domain when possible since they work better together.

- [✓ Calendar.EndDate]
- Calendar.EndTime
- Calendar.MeetingRoom
- Calendar.Message

What is Pattern?

The LUIS pattern will give more intelligence to the utterances and conversational flow in chat bot. This will improve the performance of NLP engine and analysis more deeply for the utterances sub entities information. All the entities in patterns are surrounded by curly brackets, {} also patterns can include entities, and entities with roles.

Here is the example for patterns that we have added in the LUIS application. In this “State” is the LIST entity and added inside curly brackets, {} because patterns are following some kind of template mode.

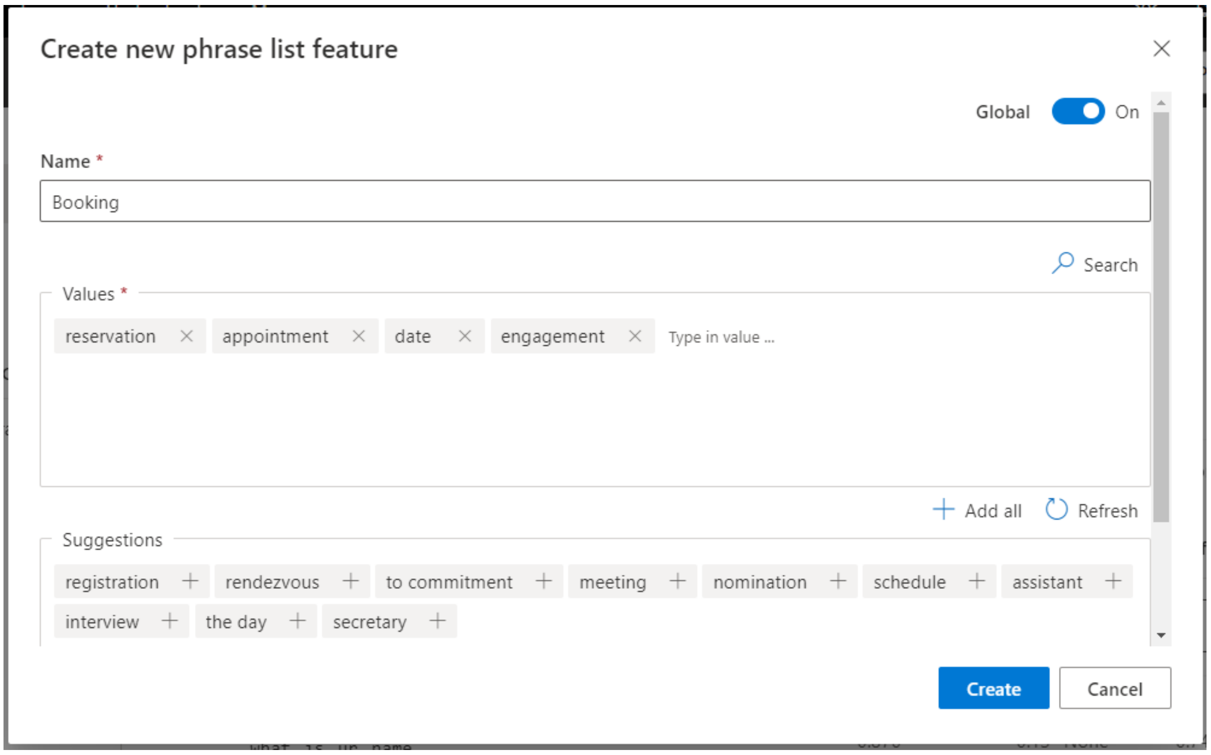

Patterns
BillPayment
Type in templates of what you expect your users to say and hit Enter.
Reassign intent
Edit
Delete
Filters

User input ↑	Intent
where is {State}?	BillPayment

What is the Phrase list in a LUIS application?

Phrase lists are similar to the synonymous or the similar meaning of the word that we have entered. This will give more intelligence to the utterances and improve the conversation flow in the LUIS application.

E.g. The real-time example for phrase list is the booking synonym, or similar meaningful words are reservation, appointment, date, engagement, etc.


Create new phrase list feature

Global: On

Name *

Booking

Values *

reservation, appointment, date, engagement

Type in value ...

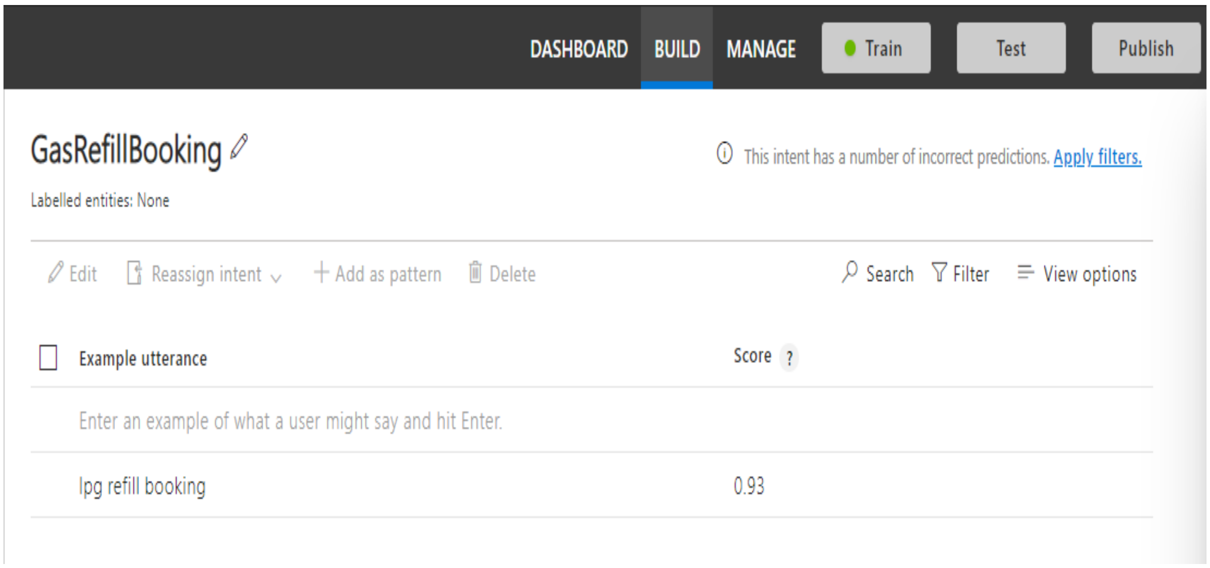
Suggestions

registration, rendezvous, to commitment, meeting, nomination, schedule, assistant, interview, the day, secretary

While adding a phrase list feature for the intent, it will give us a number of suggestions in the suggestion box with the help of machine learning features. The phrase list will be part of the machine learning feature in a LUIS application and we can add other prebuilt features as well. Once enabled, the global feature will include the global vocabulary of the synonyms.

How to Train a LUIS application?

Training should be a building process in the LUIS application and it will be keeping the app up to date, and if the app is up to date then it will be displayed as a green circle on the Train menu.



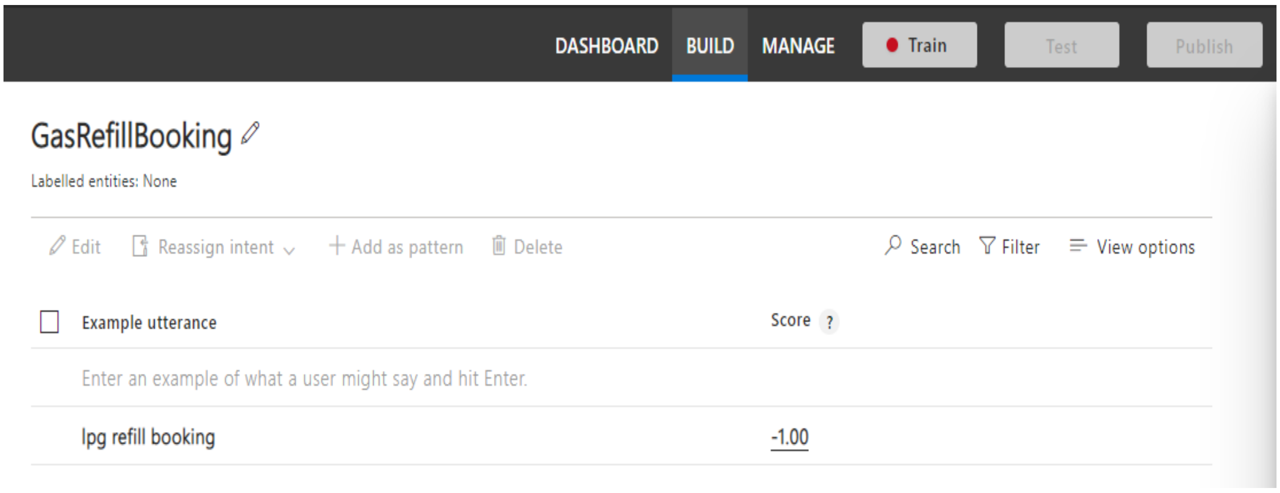
GasRefillBooking

Labelled entities: None

This intent has a number of incorrect predictions. Apply filters.

Example utterance	Score
Enter an example of what a user might say and hit Enter.	
lpg refill booking	0.93

The red circle displayed in the Train menu denotes that the app has untrained changes or some changes have happened during the interval of time which is not yet updated in the application. So, we can update the application by clicking on the Train menu button in the LUIS application and once the Train is successful then the red color changes into green. Whenever the color of the Train is negative, the score of the utterances will be negative (–1). Once build is successful, the score changes to positive (0.5,1.0...)



GasRefillBooking

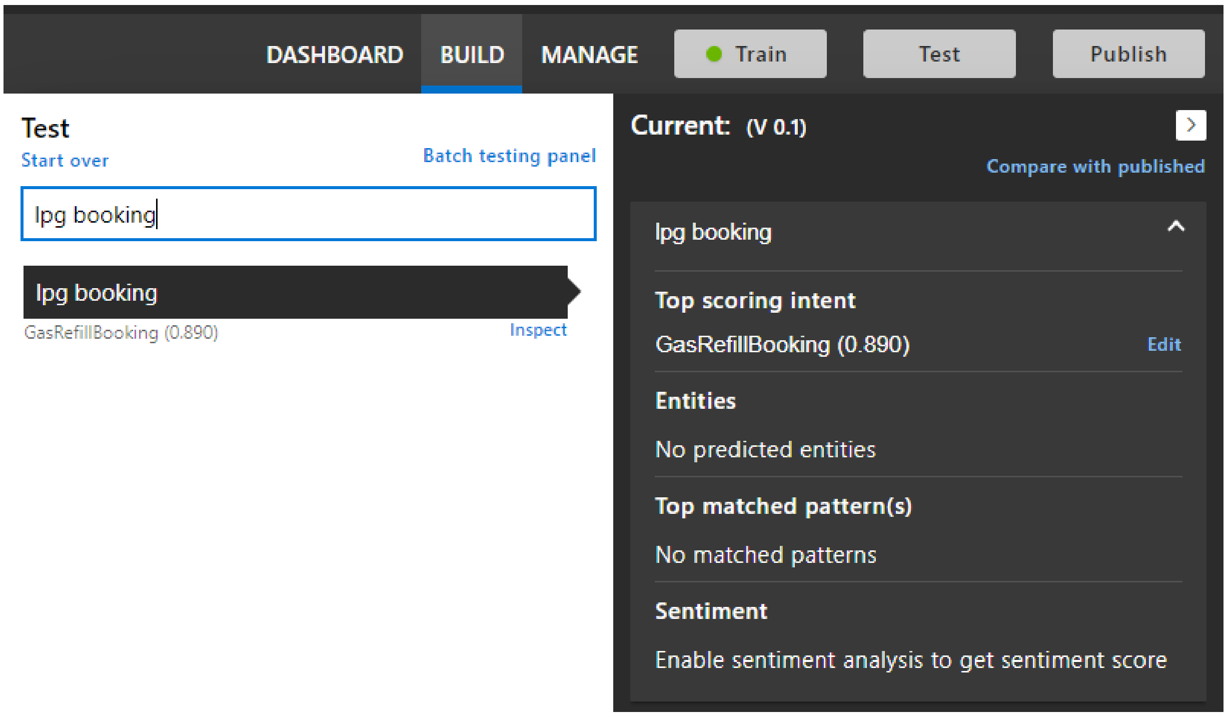
Labelled entities: None

Example utterance	Score
Enter an example of what a user might say and hit Enter.	
lpg refill booking	-1.00

Note : *"Click on the Train menu and update the app"*

How to Test a LUIS application?

Once the application build is successful or the circle in the Train menu changes to green, we can test the application through the help of a Test menu in the LUIS application. The test menu will open a test chat window and it will give you the brief details about the entered text or utterances in the chat box.



lpg booking

lpg booking

GasRefillBooking (0.890)

Current: (V 0.1)

lpg booking

Top scoring intent

GasRefillBooking (0.890)

Entities

No predicted entities

Top matched pattern(s)

No matched patterns

Sentiment

Enable sentiment analysis to get sentiment score

The start over link will reset the history of the chat and it will create or begin a new chat history.

How to Publish a LUIS application?

As we all know, the term Publish is used to make the application live. There are two types of development slots available in the Luis application.

- Staging
- Production

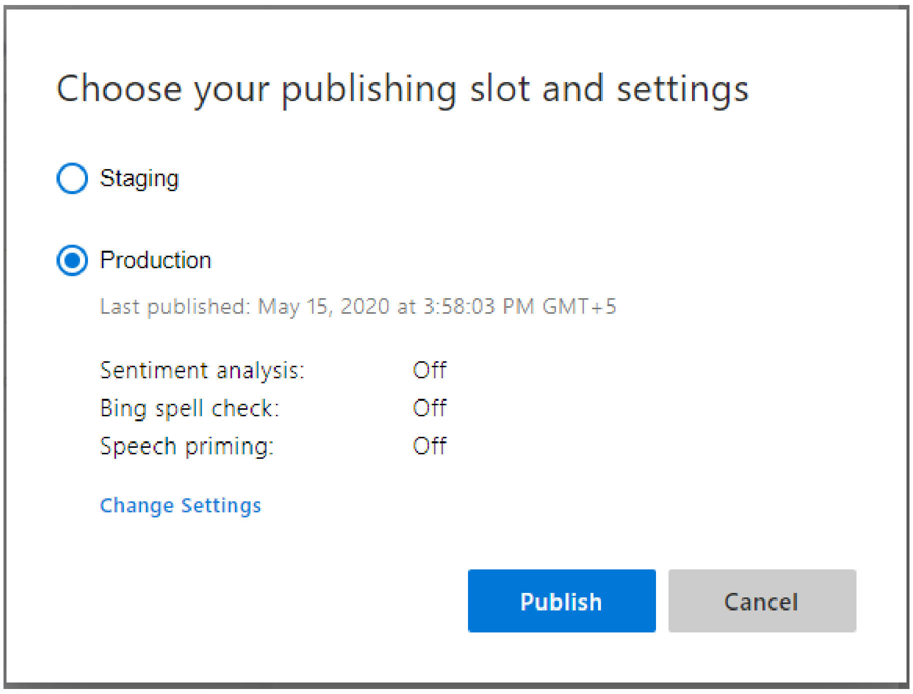
Select the appropriate development slot and publish the LUIS application to the respective environment;this will be more helpful during the development phase of an application.

Staging

The Staging environment is used for the purpose of testing only. In this way, we can improve the quality of the LUIS application or the product.

Production

Once the application is ready for use then publish on the production slot and it will be available for public use.


Choose your publishing slot and settings

[ ] Staging

[✓ Production]

Last published: May 15, 2020 at 3:58:03 PM GMT+5

Sentiment analysis:	Off
Bing spell check:	Off
Speech priming:	Off

Note: We can choose any of the development slots for publishing the LUIS application environment.

What is QnA Maker?

One of the important language category APIs in Cognitive Services is QnA Maker. It is a cloud-based API service provided by Microsoft that lets you create a conversational question and answer layer over your data. All of the chatbot's answers come from an already-configured custom KB (or Knowledge Base) in QnA Maker or any other medium like LUIS, etc. We all know most industries are using custom chatbots in their website for customer assistance to reduce additional work. In the current Covid-19 pandemic assisting large amounts of customers through any medium is quite difficult. So implementing customer assisting chatbots is relevant for this current situation, and in the future as well. There are very simple real time scenarios like leave requests and ticket creation, which are handled easily through chatbots with the help of QnA maker knowledge base, or LUIS. Nowadays many people are using QnA Maker for most chatbot conversational flow.

One of the most familiar examples for QnA Maker implementation is FAQ (Frequently asked questions) in the chatbot. If the user is asking one or more questions to an intelligent chat bot (robot or non-human agent) through an chat window that appear in the website or like Facebook messenger, slack, Microsoft teams, skype, etc. The user will get an appropriate or predefined answer from the chat bot, so all these FAQ or predefined questions are stored in the QnA maker Knowledgebase.

We can easily create a QnA Maker model with the help of a https://www.qnamaker.ai/ account and for this we require "QnA service in Microsoft Azure." For that, first we need to create a QnA service API in Azure.

What’s new for QnA Maker?

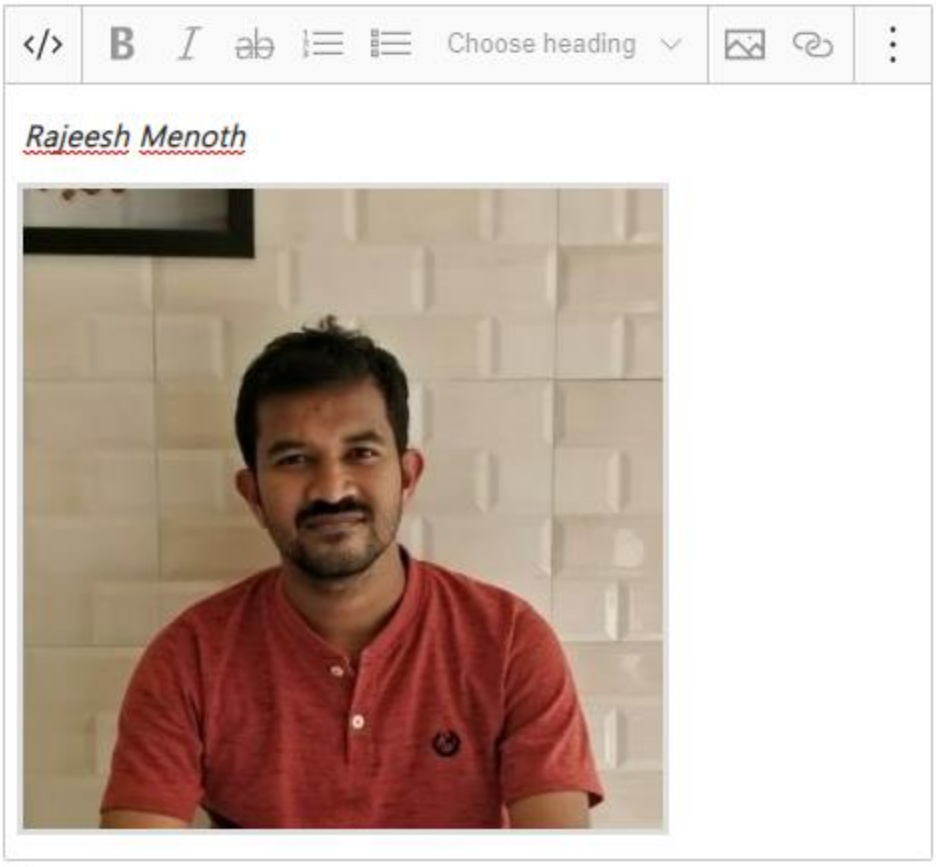
As per the latest Microsoft build in 2020, Microsoft has announced new features for QnA Maker. The following features will be helpful to handle more extra features in QnA maker applications in real time scenarios:

- Rich Text Authoring
- Role Based Access Control (RBAC)

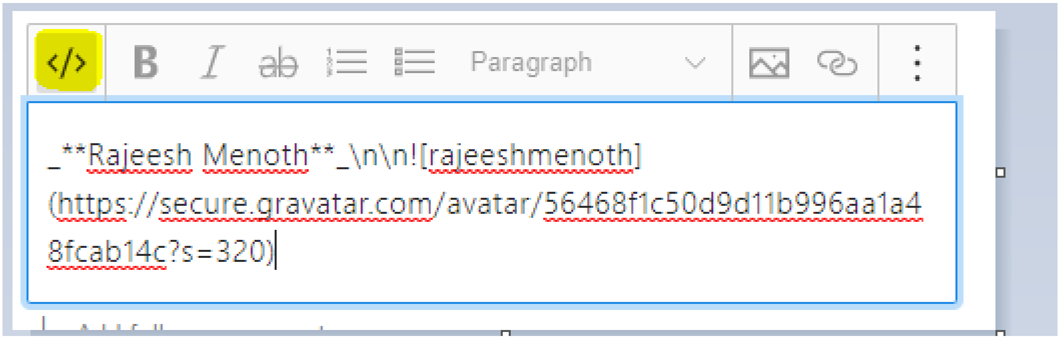
Rich Text Authoring

This is a very cool feature introduced in Microsoft build 2020 and it’s a rich text editor with relevant activity. We can add an image, video or document url in the rich editor as well as content formatting activities. By clicking on the respective image and document icon in the editor menu we can achieve the media activities. If you are facing an issue for adding a video in the editor section then just click on the “</>” menu because it will display the markdown syntax option and there you can copy paste the respective markdown syntax for videos.

E.g. This will be a simple example for a rich text editor in QnA Maker.



E.g. Clicking on the "</>" icon in the rich text editor will display the detailed markdown syntax information for the above screenshot.



```
_**Rajeesh Menoth**_\n\n![rajeeshmenoth]
(https://secure.gravatar.com/avatar/56468f1c50d9d11b996aa1a4
8fcab14c?s=320)
```

Important: The video link is required as a thumbnail image option in the editor section then it will display the video image otherwise it will display as a corrupted image in the editor view. So, we can add video markdown syntax included thumbnail as manually in the markdown section and it will work as expected.

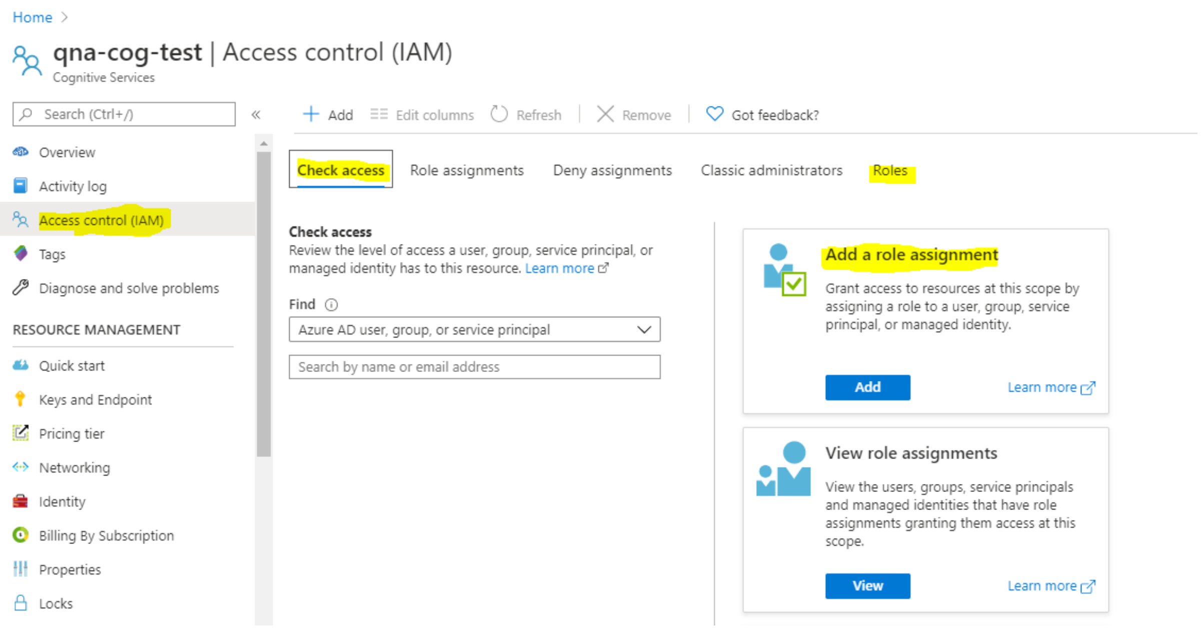
Role Based Access Control (RBAC)

Every application has three types of major roles which are reader, editor, and publisher. Microsoft has configured it this way as well. The role assign option will be available in QnA maker cognitive service in Azure portal.

Each role has the following capabilities to control the QnA maker user activities for E.g. Edit, Add, Import, Delete, Publish, etc.

Access control (IAM)

Authorizations are the major criteria for accessing an application by specific users with certain rules. We can easily apply these rules in QnA maker with the help of RBAC or Role Based Access Control in Azure and this will improve the security of accessing the application. Click on the respective QnA maker cognitive service in Azure portal that we have already created or newly created for the QnA maker activities. It will redirect you to the detailed page or overview page of cognitive services of QnA Maker. Click on the left menu called "Access control" or "IAM" and it will provide all the types of user permissions activities like Check access, Roles, etc.



Check access
Review the level of access a user, group, service principal, or managed identity has to this resource. Learn more

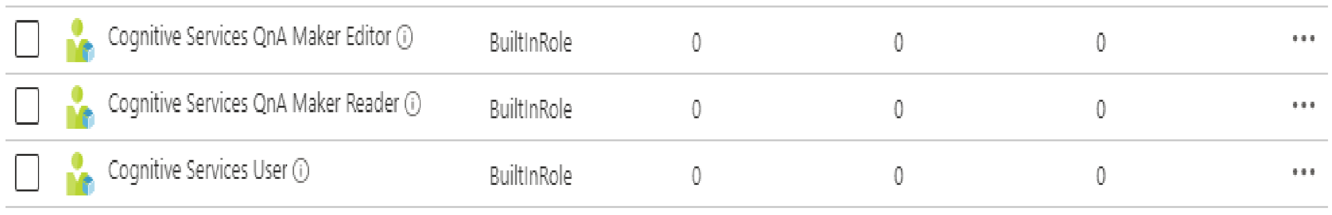
Find
Azure AD user, group, or service principal
Search by name or email address

Add a role assignment
Grant access to resources at this scope by assigning a role to a user, group, service principal, or managed identity.

View role assignments
View the users, groups, service principals and managed identities that have role assignments granting them access at this scope.

Roles

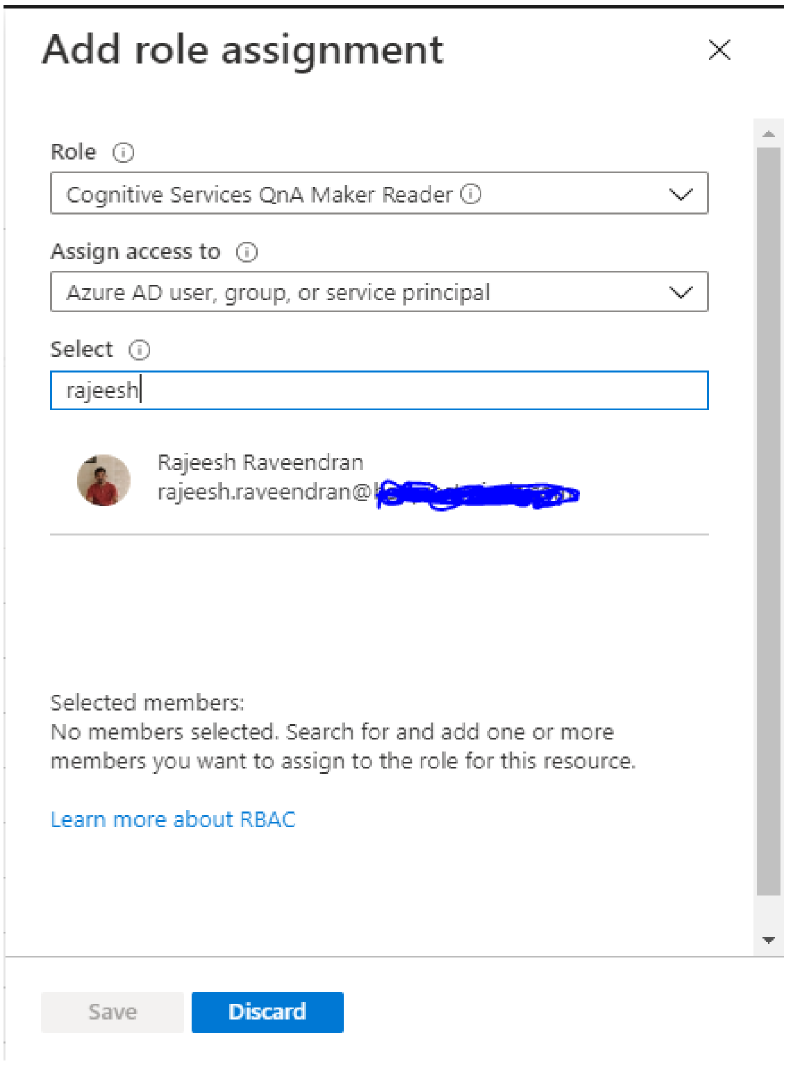
Before going to add a role assignment for user activity in QnA maker we can see the important roles that we want to use for the further activities.



☐	Cognitive Services QnA Maker Editor ⓘ	BuiltInRole	0	0	0	•••
☐	Cognitive Services QnA Maker Reader ⓘ	BuiltInRole	0	0	0	•••
☐	Cognitive Services User ⓘ	BuiltInRole	0	0	0	•••

Add a role assignment

Click on the "Add a role assignment" button in "Check access" menu tab and search the respective role, here, we have selected Reader as the role so they can only read QnA maker information like Testing activity for the QA team.



Add role assignment

Role ⓘ

Cognitive Services QnA Maker Reader ⓘ

Assign access to ⓘ

Azure AD user, group, or service principal

Select ⓘ

rajeesh

Rajeesh Raveendran
rajeesh.raveendran@[redacted]

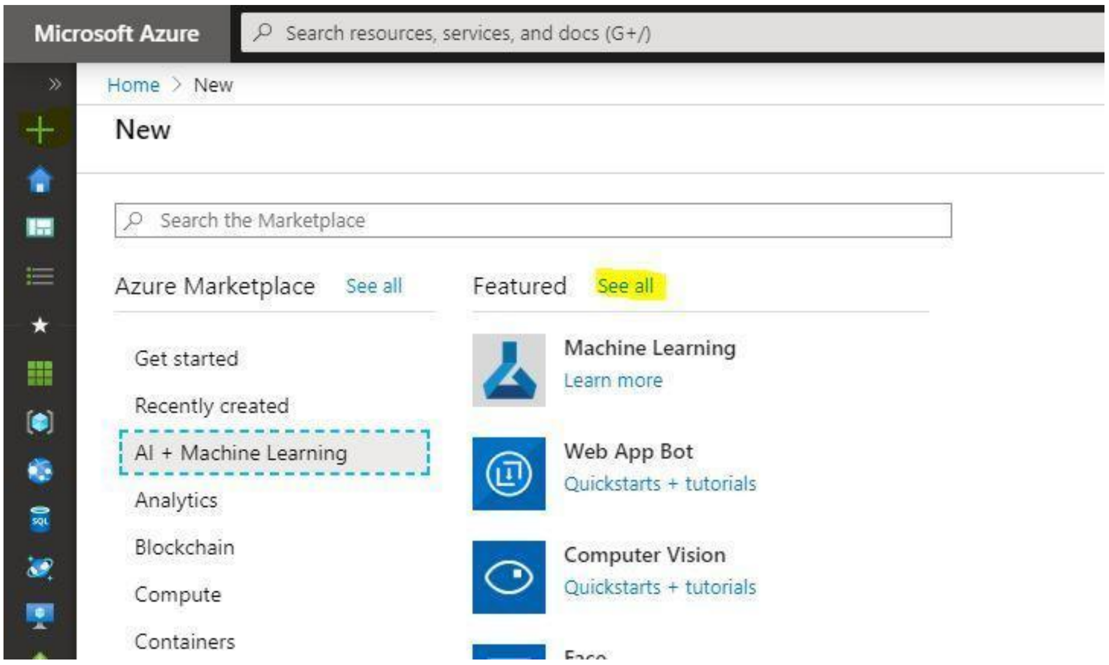
Selected members:
No members selected. Search for and add one or more members you want to assign to the role for this resource.

Learn more about RBAC

Create QnA Maker Service in Azure

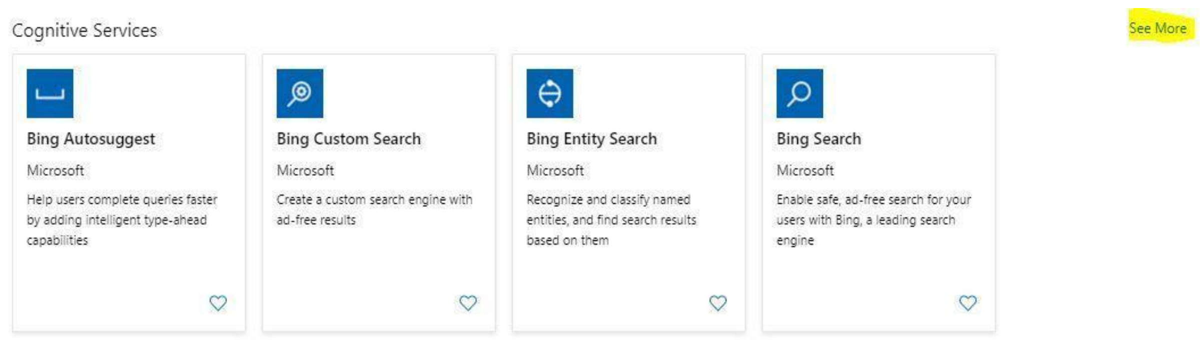
Step 1:

Go to Azure portal and click on **"+"** icon -> go to **"AI + Machine Learning"** -> Click on **"See all"**.



Step 2:

If you are not able to see LUIS in the "Cognitive Services" then go to click on the "See more" link on the right side.


Cognitive Services

See More

Bing Autosuggest
Microsoft
Help users complete queries faster by adding intelligent type-ahead capabilities

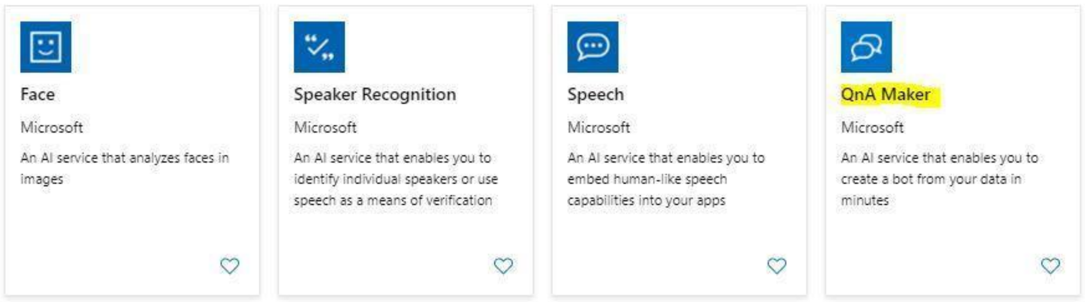
Bing Custom Search
Microsoft
Create a custom search engine with ad-free results

Bing Entity Search
Microsoft
Recognize and classify named entities, and find search results based on them

Bing Search
Microsoft
Enable safe, ad-free search for your users with Bing, a leading search engine

Step 3:

Once you are able to see QnA Maker in the "Cognitive Services" list then go and click on it.


Face
Microsoft
An AI service that analyzes faces in images

Speaker Recognition
Microsoft
An AI service that enables you to identify individual speakers or use speech as a means of verification

Speech
Microsoft
An AI service that enables you to embed human-like speech capabilities into your apps

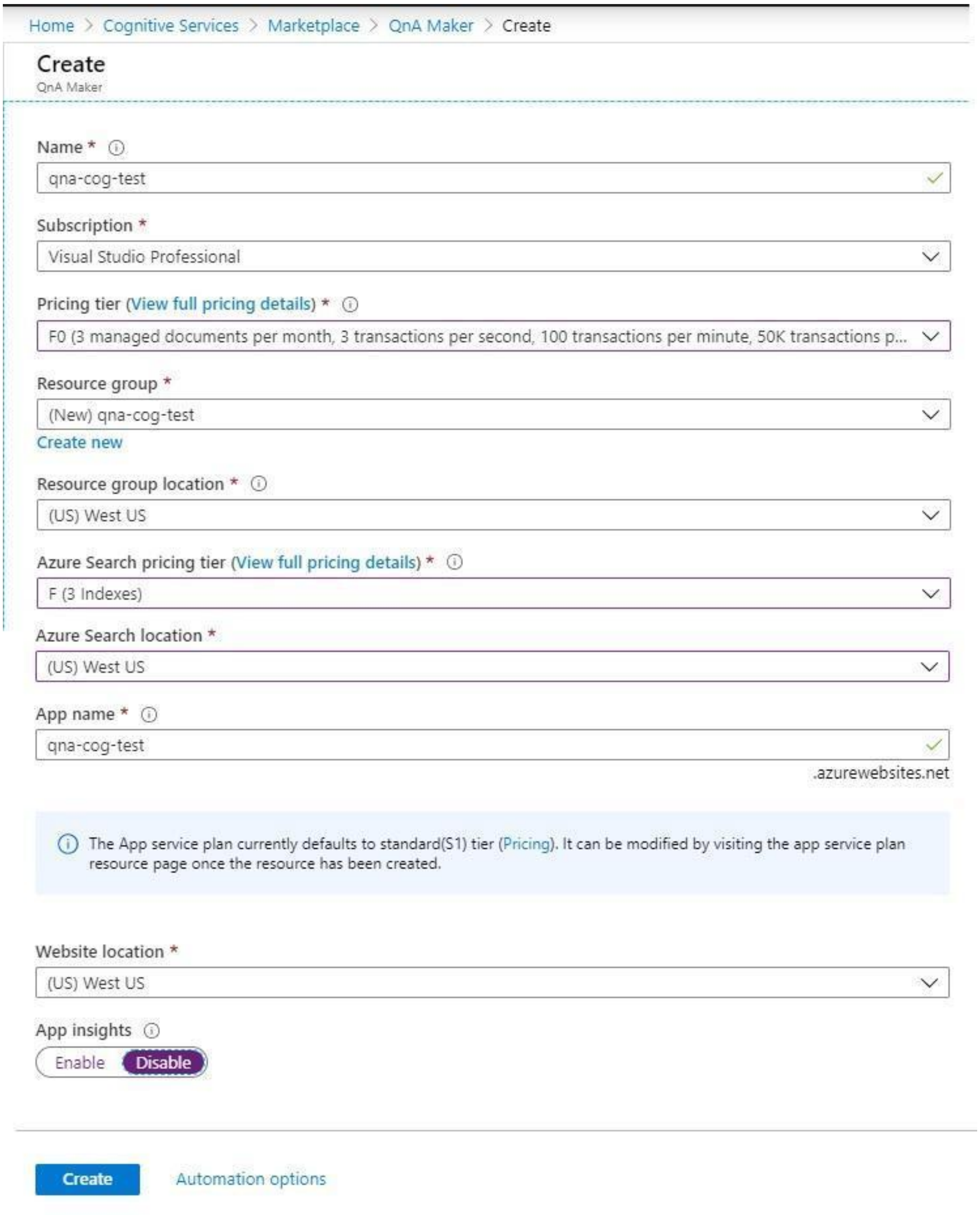
QnA Maker
Microsoft
An AI service that enables you to create a bot from your data in minutes

Step 4:

The following screen will appear once you click on "Create" button in "QnA Maker"

- **Name:** We can give one unique name for our QnA Maker service.
- **Subscription:** We can select our Azure subscription for QnA Maker service.
- **Pricing tier:** As of now there are two pricing tiers available, "FO" & "SO", and obviously "FO" is the free one. We can choose the appropriate pricing tier as per our needs. When you are planning to do performance testing then choose “SO” as the priority one because it will handle more number of hits per second otherwise choose “FO” more for learning purposes.
- **Resource group:** We can create a new resource group or choose from an existing one (We created a new resource group as "qna-cog-test").
- **Resource group location:** We can select our location from the Resource group location. The best thing is we can choose a location closest to our customer's needs.
- **Azure search pricing tier:** Each index will be considered as Knowledge Base publish permission, so based on the needs select the appropriate index accordingly.
- **Azure search location:** The best thing is we can choose a location closest to our customer needs.
- **App name:** This will be part of our service url and give proper name.
- **Website location:** The best thing is we can choose a location closest to our customer needs.

- **App insights:** If you are planning to track the service and log , etc then you should enable app insights.


Home > Cognitive Services > Marketplace > QnA Maker > Create

Create
QnA Maker

Name * : qna-cog-test

Subscription * : Visual Studio Professional

Pricing tier (View full pricing details) * : F0 (3 managed documents per month, 3 transactions per second, 100 transactions per minute, 50K transactions p...

Resource group * : (New) qna-cog-test
Create new

Resource group location * : (US) West US

Azure Search pricing tier (View full pricing details) * : F (3 Indexes)

Azure Search location * : (US) West US

App name * : qna-cog-test
.azurewebsites.net

The App service plan currently defaults to standard(S1) tier (Pricing). It can be modified by visiting the app service plan resource page once the resource has been created.

Website location * : (US) West US

App insights: Enable [✓ Disable]

- Click on the "Create" button and wait for the build to succeed.
- Once the build has succeeded, then click on the "Dashboard" and we can see "qna-cog-test" is created in the all resources list. QnA is ready for use!

Create Knowledge Base in QnA Maker

We have already created the QnA service in Microsoft Azure and now we can easily create a Knowledge Base in our QnA Maker account. So go to QnA Maker account and create a new Knowledge Base.

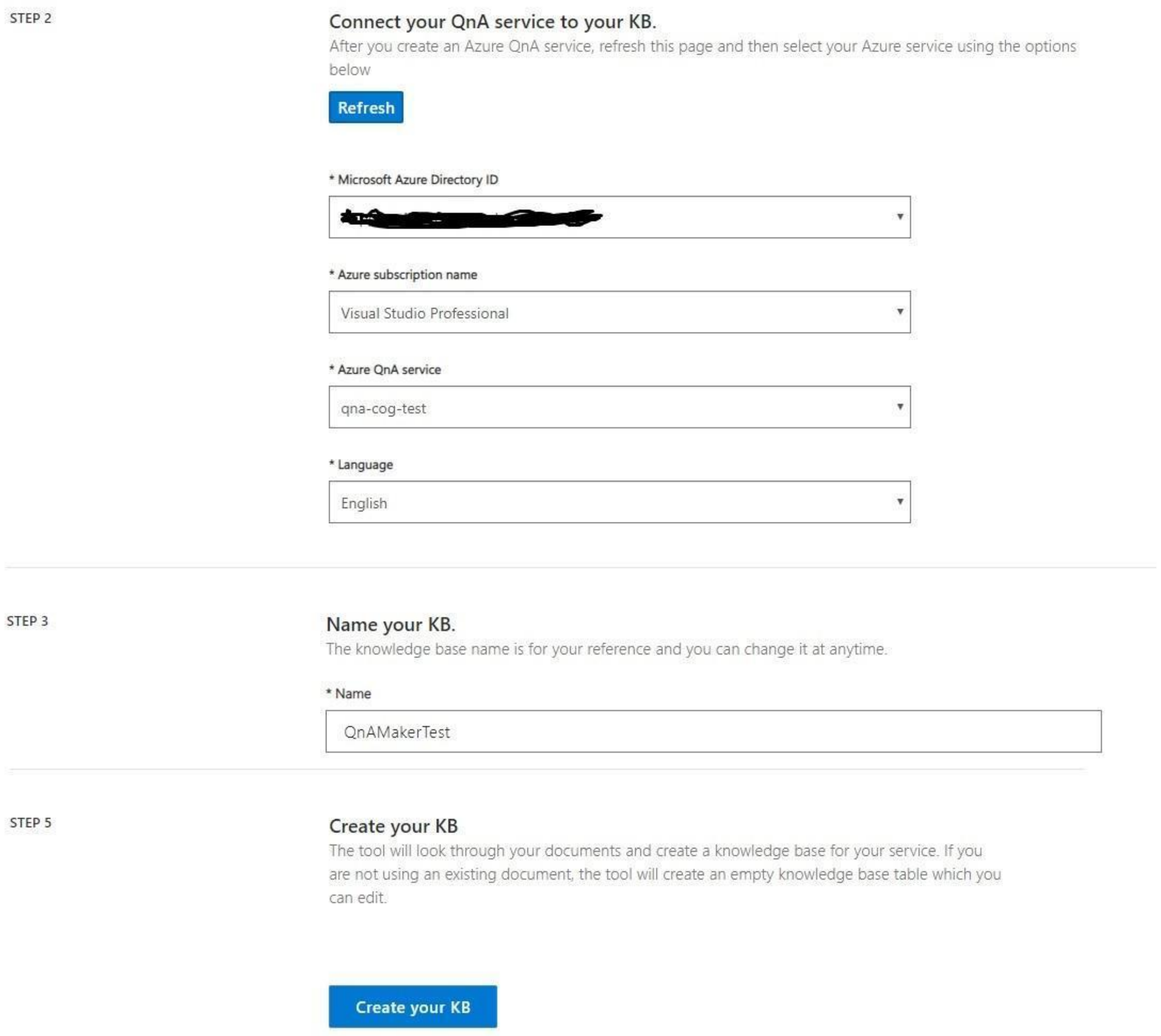
Microsoft Azure Directory ID: Select the appropriate directory from the list.

Azure subscription name: We can select our Azure subscription for QnA Maker service.

Azure QnA service: The QnA service that we have created in azure.

Language: The current culture or language we're going to use in QnA Maker application.

Name your KB: The name of your KB.


STEP 2

Connect your QnA service to your KB.
After you create an Azure QnA service, refresh this page and then select your Azure service using the options below

Refresh

* Microsoft Azure Directory ID
[redacted]

* Azure subscription name
Visual Studio Professional

* Azure QnA service
qna-cog-test

* Language
English

STEP 3

Name your KB.
The knowledge base name is for your reference and you can change it at anytime.

* Name
QnAMakerTest

STEP 5

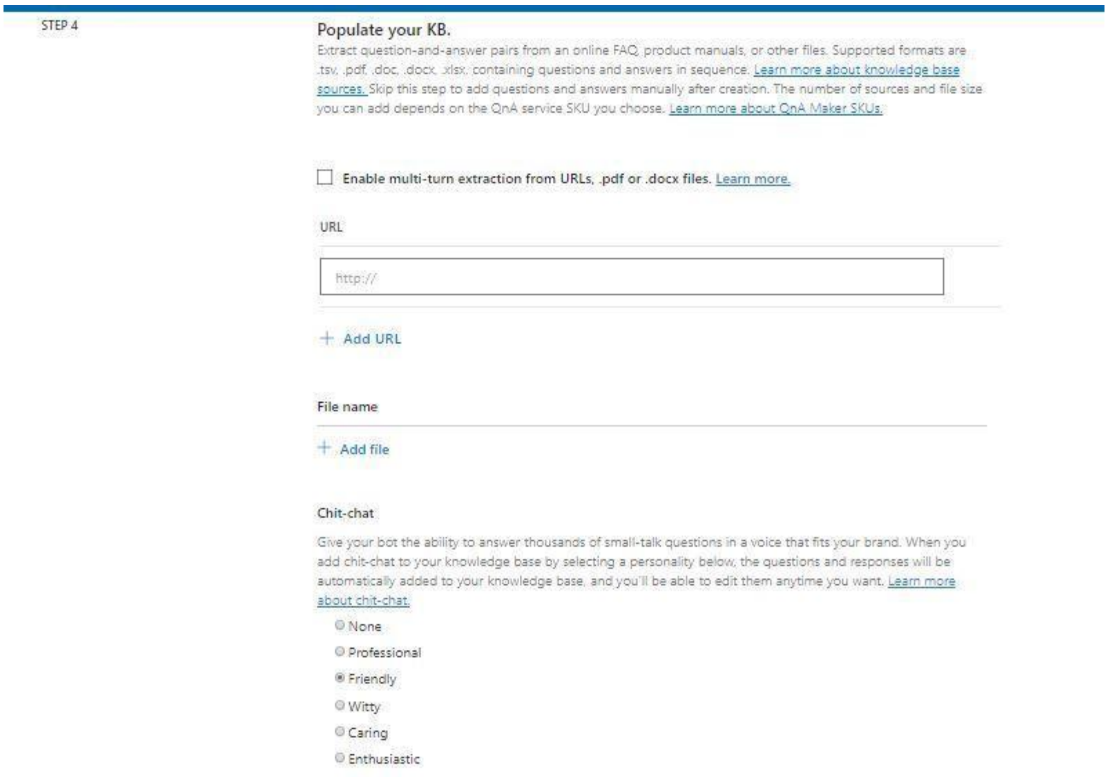
Create your KB
The tool will look through your documents and create a knowledge base for your service. If you are not using an existing document, the tool will create an empty knowledge base table which you can edit.

Create your KB

- Create your KB.

Step 4:

The following step is not mandatory to add. If you select any chit-chat options then automatically generate custom questions and answers in the respective Knowledge Base.


STEP 4

Populate your KB.

Extract question-and-answer pairs from an online FAQ, product manuals, or other files. Supported formats are .tsv, .pdf, .doc, .docx, .xlsx, containing questions and answers in sequence. Learn more about knowledge base sources. Skip this step to add questions and answers manually after creation. The number of sources and file size you can add depends on the QnA service SKU you choose. Learn more about QnA Maker SKUs.

☐ Enable multi-turn extraction from URLs, .pdf or .docx files. Learn more.

URL

http://

+ Add URL

File name

+ Add file

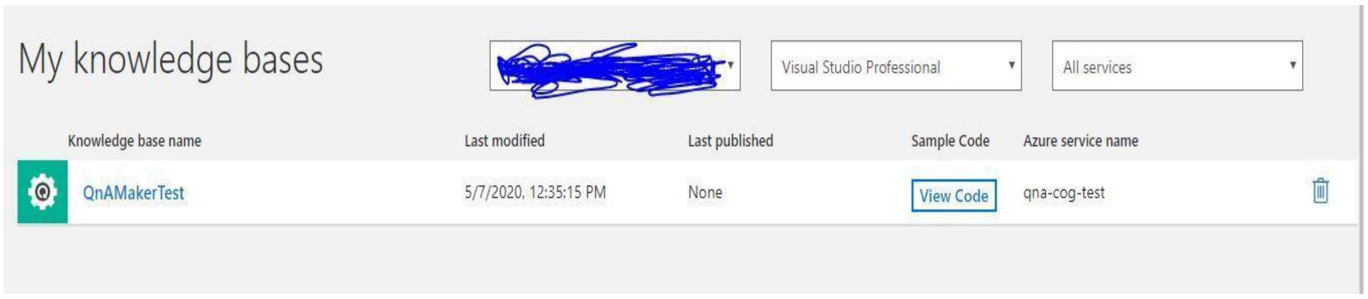
Chit-chat

Give your bot the ability to answer thousands of small-talk questions in a voice that fits your brand. When you add chit-chat to your knowledge base by selecting a personality below, the questions and responses will be automatically added to your knowledge base, and you'll be able to edit them anytime you want. Learn more about chit-chat.

○ None
○ Professional
◉ Friendly
○ Witty
○ Caring
○ Enthusiastic

Output:

The Knowledge Base is successfully created in QnA Maker with friendly chit-chat questions and answers.


My knowledge bases

[redacted] | Visual Studio Professional | All services

Knowledge base name	Last modified	Last published	Sample Code	Azure service name
QnAMakerTest	5/7/2020, 12:35:15 PM	None	View Code	qna-cog-test

What is Chit-Chat?

Chit-Chat contains prebuilt chit-chat questions in the knowledge base in QnA Maker. This will be a very simple option to integrate a pre-populated set of the top chit-chat or questions in the initial creation of an empty knowledge base. Adding Chit-Chat in the knowledge base will improve and engage the chatbot conversational flow as more effective.

We can choose an appropriate Chit-Chat option other than “None” (Empty knowledge base) during the creation of knowledge base in QnA Maker.

Types of Chit-Chat in Knowledge Base

The following are the currently available Chit-Chat options in the knowledge base and based on the question **“When is your birthday?”** each chit-chat option will respond with various types of answers.

Personality	**Example**
Professional	Age doesn't really apply to me.
Friendly	I don't really have an age.
Witty	I'm age-free.
Caring	I don't have an age.
Enthusiastic	I'm a bot, so I don't have an age.

E.g. The Professional Chit-Chat option will provide all kinds of professional answers and we can choose this option for the business conversational flow. We can try a friendly Chit-Chat option for the customer interaction in chatbot conversation flow.
Note: All chit chat selections are based on our business requirements.

Type of Language Support

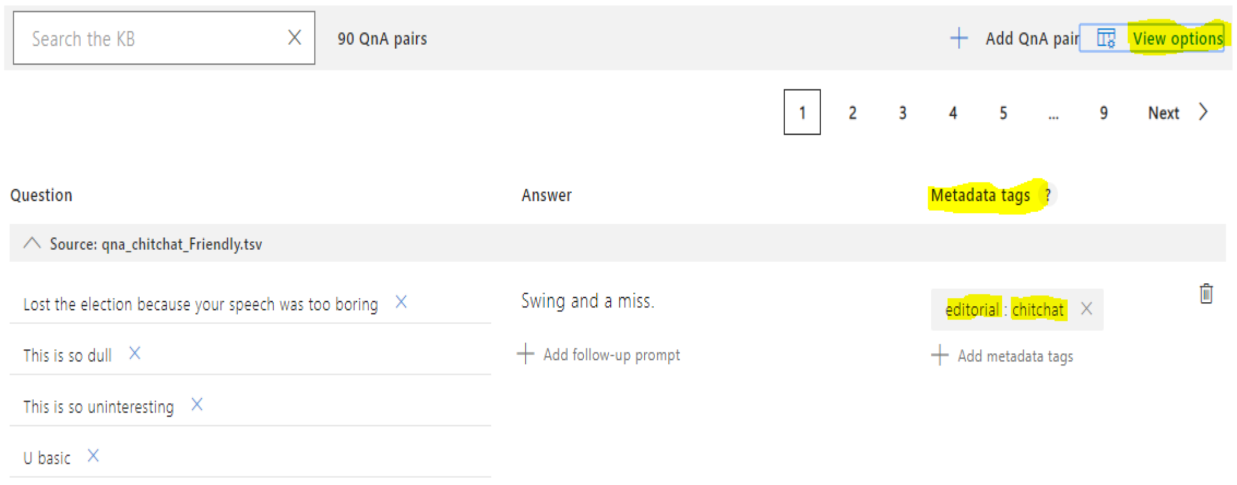
Chit-chat data sets are supported in the following languages.

- Chinese
- English
- French
- Germany
- Italian
- Japanese
- Korean
- Portuguese
- Spanish

Metadata in QnA Maker

The definition of a Metadata is data about the data and in knowledge base it will contain important information about the question pair such as store product id , group name, etc as Metadata in the question pair. Metadata is a text-based Key & Value pair used in each and every question in the knowledge base. It is not mandatory to add metadata in the question section and also is case sensitive in nature and it will store all the Key & Value info into small letters or lowercase.

Metadata information by default hides from the editorial page in the knowledge base and enables it by clicking on the “Show Metadata” menu in the “View options” menu list present in the knowledge base. It will list already added or empty Metadata tags in each and every question pair. The following example “editorial” is the Key and “chitchat” is the value of Metadata and it will always store it as a small letter.


Search the KB
90 QnA pairs
Add QnA pair
View options
1 2 3 4 5 ... 9 Next

Question	Answer	Metadata tags
Source: qna_chitchat_Friendly.tsv		
Lost the election because your speech was too boring	Swing and a miss.	editorial : chitchat
This is so dull	Add follow-up prompt	Add metadata tags
This is so uninteresting		
U basic		

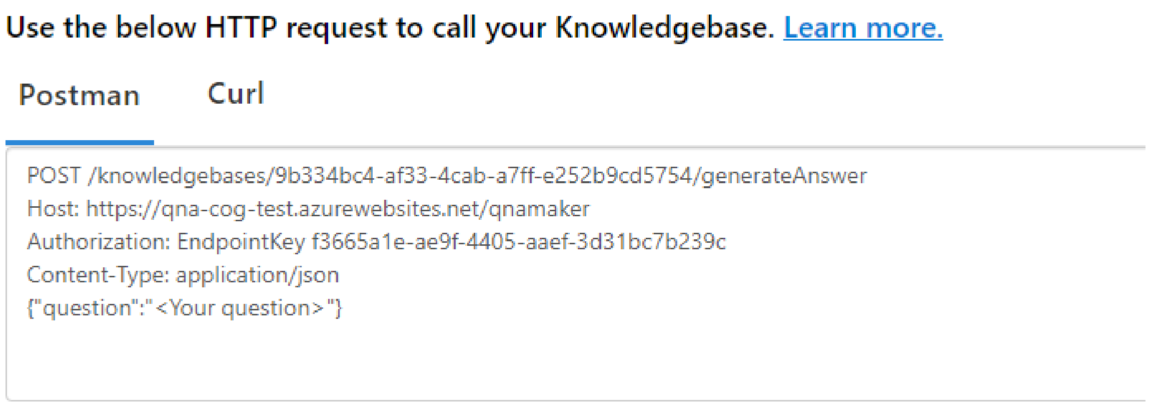
These are important points to keep in mind while creating a metadata in the knowledge base.

- Length of metadata key/value text: 100
- Supported characters for metadata name: Alphabets, digits and _
- Supported characters for metadata value: All except **:** and **|**

Generate search API in QnA Maker

We can expose all the knowledge base information using cognitive service api. Once published (click on the PUBLISH menu in KB similar to what we have done in LUIS application) then it will automatically generate the following information to call a knowledge base through an HTTP request.

The combination of “Host” + “POST” is the endpoint URL of search API, Authorization and Content-Type is part of the header parameter. The value should be part of the request body as JSON format type.



Use the below HTTP request to call your Knowledgebase. Learn more.

Postman Curl

```
POST /knowledgebases/9b334bc4-af33-4cab-a7ff-e252b9cd5754/generateAnswer
Host: https://qna-cog-test.azurewebsites.net/qnamaker
Authorization: EndpointKey f3665a1e-ae9f-4405-aaef-3d31bc7b239c
Content-Type: application/json
{"question":"<Your question>"}
```

Knowledge Base API response

The generated search API credentials can be used for accessing the knowledge base information. Based on the requirement we can configure additional information to the API request.

We can include the following parameters in the request body based on the requirements.

```
{
        "question":"How are you?",
        "isTest": false,
        "Top": 2
}
```

question: The question that we are going to ask.

IsTest: This Boolean property will decide whether it is a production or testing environment. If “isTest” is false then it denotes production environment and will get published in knowledge base information in api search. The true flag will be the exact opposite of a production environment, that is, it would be Test environment.

Top: This will return the top question pair in the response based on the threshold score in QnA maker. E.g. Top 2 questions will return from the QnA maker based on the score.

Note: The question parameter must require in the api request body.

Output:

The following output returns based on the question that we have given in the question parameter. That is the only parameter we have considered in the request body and other parameters are not mandatory to add in the request body section.

```
{
   "answers": [
      {
         "questions": [
            "Hey, how are you?",
            "Are you doing good?",
            "How are you feeling?",
            "How are you doing?",
            "How are ya?",
            "How are things?",
            "How are you?",
            "How are you going?",
            "How art thou?",
            "Greetings, Bot. How are you doing?",
            "Are you feeling well?",
            "Are you feeling OK?",
            "Are you feeling good?",
            "Are you doing well?",
            "Are you doing OK?",
            "How are things going?",
            "How's it going?",
            "Say, how are you doing?",
            "How's tricks?",
            "How's the day treating you?",
          ],
          "answer": "I'm doing great, thanks for asking!",
          "score": 100,
          "id": 58,
          "source": "qna_chitchat_Friendly.tsv",
          "metadata": [
            {
              "name": "editorial",
              "value": "chitchat"
            }
          ],
          "context": {
            "isContextOnly": false,
            "prompts": []
          }
        }
    ],
    "debugInfo": null,
    "activeLearningEnabled": false
}
```

If the question that we have asked is in any of the knowledge bases in QnA maker then it will return the detailed information about that particular question or qna pair. We can return multiple qna or question pairs for giving the “Top” parameter in the request body.

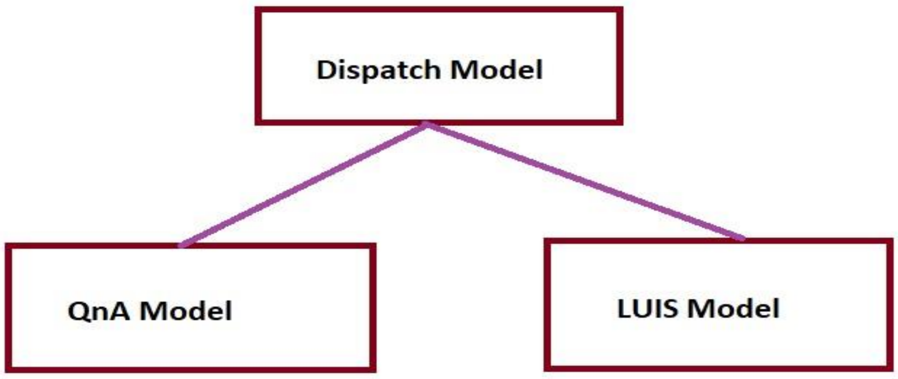
Summary of Dispatch

As we know, all of the applications that we are using in the real world are growing day by day. So, handling all of this single handedly is very difficult in the current situation since it will take too much time to develop and maintain the application even if you are using a chatbot application in a real-time scenario. As a Microsoft chatbot developer we are familiar with QnA maker knowledge base and Luis models for the chatbot e2e communication. For E.g. all the content that we have entered into the chatbot is considered as utterances in the Luis model and questions in the knowledge base. When we are going to use multiple models in our application at a time, we need to handle, train and publish the model and we also need to hit multiple api calls for the relevant response for each activity. So, we can reduce all this activity using the dispatch command tool in the npm package. The dispatch has the capability to handle multiple bot modules for the e2e communication. This will be very simple and we can set this up as a web job or manual command execution in our local system, virtual machine, CI / CD and Azure app service as well.

Dispatch Command Line Tool

The name itself says it is dispatching something, yes this will be helpful when we are using multiple models in our application. *"The dispatch tool to create and evaluate LUIS models is used to dispatch intent across multiple bot modules such as LUIS models, QnA knowledge bases and others (added to dispatch as a file type)"*.

In the above description it is clearly mentioned that you can add both LUIS and QnA Maker models together in dispatch LUIS App or create any of the models. The dispatch App has the capability to handle multiple bot modules together without impacting the performance of the application. The dispatch LUIS App will find out the intent based on the given utterances across all the modules that we have integrated in the dispatch version of the LUIS application.



Note: The main purpose of the dispatch app is to identify and route intent for multiple bot modules so it is concerned only with intent classification. We cannot directly transfer entities in the dispatch application so we can choose a pattern for the entity transfer.

The Prerequisite for Dispatch tool

- Download and Install: https://nodejs.org/en/ (requires Node.js version 8.5 or above)
- Create at least one knowledge base in https://www.qnamaker.ai/
- Valid Luis account is required https://www.luis.ai/home

The command using in Dispatch tool

The following are the important commands that are used in the dispatch model creation in the LUIS application. All the descriptions are based on the document provided by dispatch in GitHub account.

Option	Description
-t or –type	luis , qna , file
-I or –id	Required only if type is luis/qna , LUIS app id or QnA kb id - from application settings page
--luisAuthoringKey	(optional) LUIS authoring key
--luisAuthoringRegion	(optional) LUIS authoring region
-s, --secret	(optional) .bot file secret
-c, --culture	(optional) Used to set LUIS app culture for dispatch. Required if none of dispatch source(s) is LUIS app.
--hierarchical	(optional) Default to true. If false, existing intents from source LUIS model(s) will be available as the dispatch intents.
-n or –name	LUIS app name or QnA name (from application settings page) or module/file name for file type
--includePrompts	(optional for QnA only) Default to false. If set to true, QnA KB prompt questions will be included in the training set
--includeMetadata	(optional for QnA only) Default to false. If set to true, QnA KB metadata will be included in the training set
-f or –filePath	Required only if type is file, Path to tsv file containing tab delimited intent and utterance fields or .txt file with an utterance on each line
--dispatch (optional)	Path to .dispatch file
--dataFolder (optional)	Dispatch working directory

-h or –help	Output usage information
-v, --version	(Required only if type is luis) LUIS app version
-i, --id	(required only if type is luis/qna) LUIS app id or QnA kb id from application settings page
-c, --culture	(optional) Used to set LUIS app culture for dispatch. Required if none of dispatch source(s) is LUIS app
-s, --secret	(optional) Secret used to encrypt/decrypt .bot file
--gov	(optional) Set to true to target Azure goverment
--remote	(optional) Set to true if invoking tool remotely

Dispatch Package Installation

First of all we need to install the "node js" (requires Node.js version 8.5 or above) package in the target machine otherwise we cannot install the "botdispatch" package because it will work on top of npm. In the prerequisite we have already installed the node.js and this command will verify the version details of node in the target machine.

node -v or node --version

After the installation of node js run this command to Install the botdispatch package in target machine.

npm install -g botdispatch

Initialization Dispatch model

The command will initialize the dispatch LUIS application and that is going to create a LUIS application with contains of all types chatbot models like qna maker, knowledge base, and LUIS models.

Command

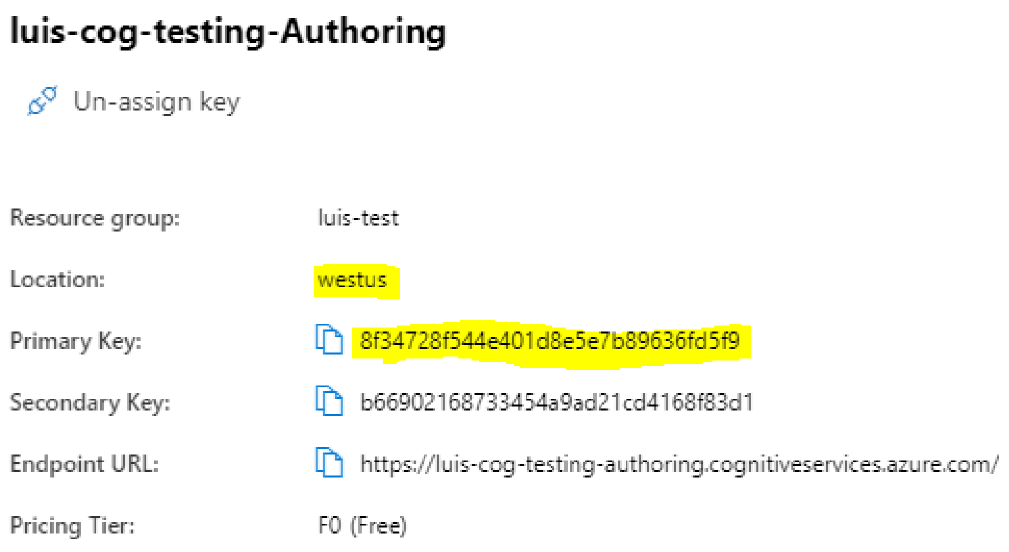
dispatch init -n <filename-to-create> --luisAuthoringKey "<your-luis-authoring-key>" --luisAuthoringRegion <your-region>

<filename-to-create>: The name of the dispatch version of the LUIS application.

<your-luis-authoring-key>: The LUIS authoring key is part of the settings section in the LUIS application.

<your-region>**:** The region of the LUIS application is part of the settings section in the LUIS application.

We can copy the region and luis authoring key from the settings section in LUIS application https://www.luis.ai/user/settings



luis-cog-testing-Authoring

Un-assign key

Resource group:	luis-test
Location:	westus
Primary Key:	8f34728f544e401d8e5e7b89636fd5f9
Secondary Key:	b66902168733454a9ad21cd4168f83d1
Endpoint URL:	https://luis-cog-testing-authoring.cognitiveservices.azure.com/
Pricing Tier:	F0 (Free)

Tips: Create the dispatch file in a specific folder or directory in your target machine because it will generate multiple files and folders for the respective intent in the dispatch luis application.

Dispatch Luis App Initialization with parameters

The initialization command will create the ".dispatch" file in the target machine. The command will contain all the relevant information about the Luis application that we have created and follow the existing steps for more clarity.

```
dispatch init –n luis-tst --luisAuthoringKey
“8f34728f544e401d8e5e7b89636fd5f9" --luisAuthoringRegion westus
```

Output: The initialization command will create the ".dispatch" file with fewer details because this will grow up while adding multiple chatbot models in the dispatch application. Now this file will contain the app id of the LUIS application.

```
{
  "authoringRegion": "westus",
  "hierarchical": true,
  "useAllTrainingData": false,
  "dontReviseUtterance": false,
  "copyLuisData": true,
  "normalizeDiacritics": true,
  "services": [],
  "serviceIds": [],
  "authoringKey": "8f34728f544e401d8e5e7b89636fd5f9",
  "version": "Dispatch",
  "region": "westus",
  "type": "dispatch",
  "name": "luis-tst"
}
```

Add QnA models in Dispatch

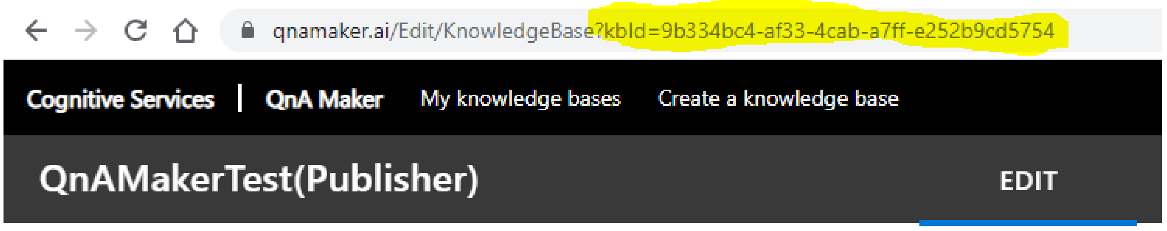
Once after initialization of the dispatch model then we can add multiple models in the Dispatch LUIS application. The command will be used for adding the QnA knowledge base models into the dispatch application using the dispatch tool.

Command for qna

```
dispatch add -t qna -i "<knowledge-base-id>" -n "<knowledge-base-name>" -k "<azure-qna-service-key1>" --intentName <intent-name>
```

The detailed information of the parameters is used in the QnA adding command in the dispatch tool.

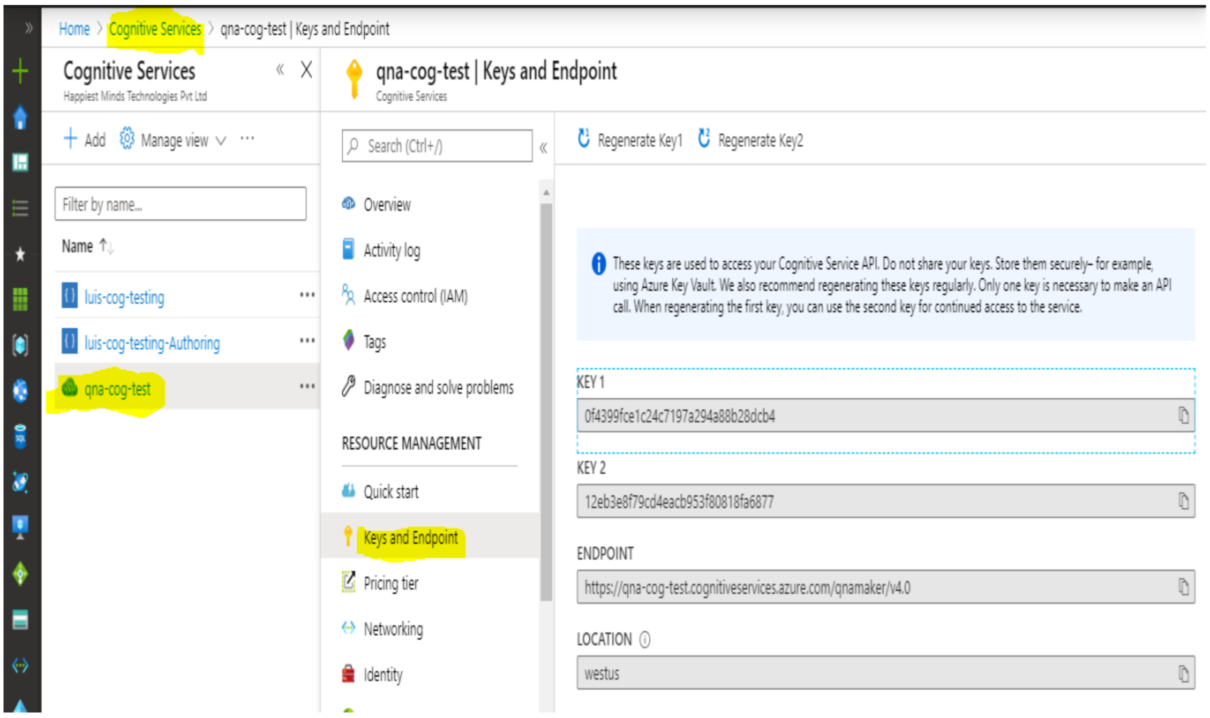
knowledge-base-id : copy from respective qna



qnamaker.ai/Edit/KnowledgeBase?kbId=9b334bc4-af33-4cab-a7ff-e252b9cd5754

knowledge-base-name: copy from respective qna

azure-qna-service-key1: copy from respective Azure qna cognitive service the key will be also called as “Ocp-Apim-Subscription-Key”.



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.

KEY 1: 0f4399fce1c24c7197a294a88b28dcb4

KEY 2: 12eb3e8f79cd4eacb953f80818fa6877

ENDPOINT: https://qna-cog-test.cognitiveservices.azure.com/qnamaker/v4.0

LOCATION: westus

intent-name: As we mentioned, the dispatch will find the intent based on the utterances that we are searching in the dispatch-created LUIS application. The intent name should be required while adding any models in the LUIS application.

This command will update an already created ".dispatch" file with qna maker information and also remember all this information now added in the target machine ".dispatch" file and not in the LUIS application.

dispatch add -t qna -i "9b334bc4-af33-4cab-a7ff-e252b9cd5754" -n "qna-cog-test" -k " 32c3d00a996d4082b0ecff2b98ccb255" --intentName q_test-qna

Finally the qna information is added in the ".dispatch" file in the target machine and here is the updated ".dispatch" file.

```
{
  "authoringRegion": "westus",
  "hierarchical": true,
  "useAllTrainingData": false,
  "dontReviseUtterance": false,
  "copyLuisData": true,
  "normalizeDiacritics": true,
  "services": [
  {
    "intentName": "q_test-qna",
    "includePrompts": true,
    "kbId": "9b334bc4-af33-4cab-a7ff-e252b9cd5754",
    "subscriptionKey": "0f4399fce1c24c7197a294a88b28dcb4",
    "type": "qna",
    "name": "qna-cog-test",
    "id": "1"
  }
 ],
 "serviceIds": [
  "1"
 ],
 "authoringKey": "8f34728f544e401d8e5e7b89636fd5f9",
 "version": "Dispatch",
 "region": "westus",
 "type": "dispatch",
 "name": "luis-tst"
}
```

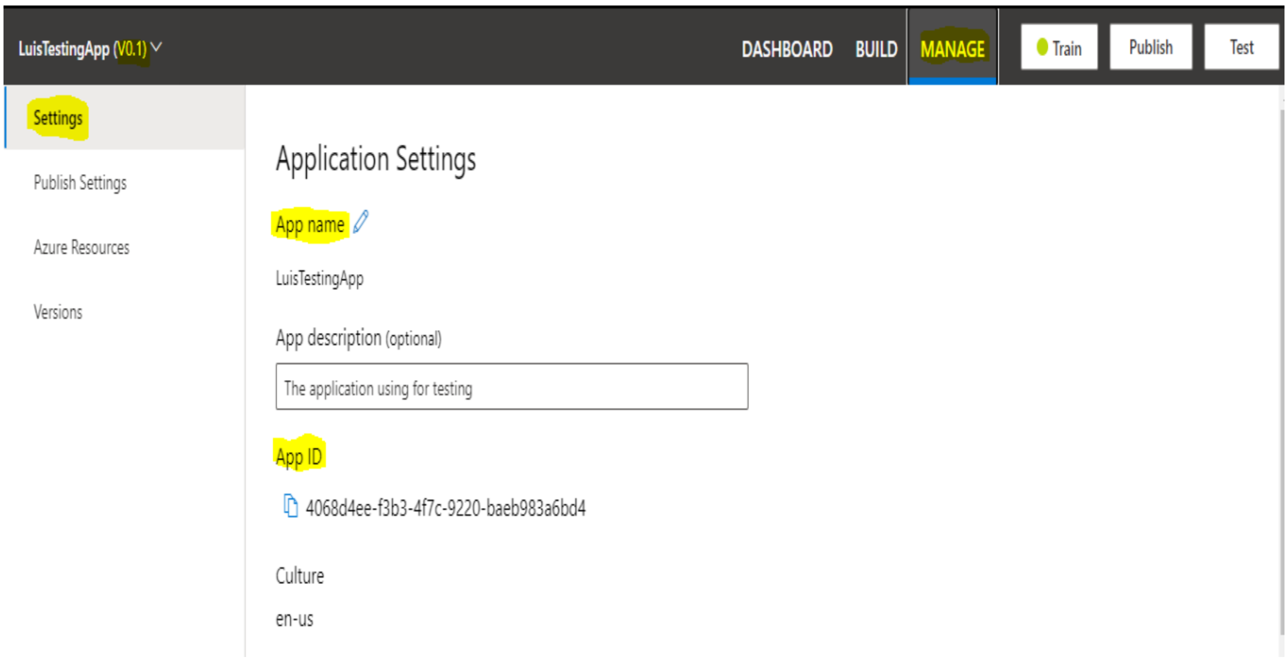
Note: The model will be growing day by day and in the multiple model identification we can add a prefix in each intent in the dispatch model LUIS application. Here we have added the prefix “q” for QnA maker and “l” for LUIS (manually created luis model).

Add LUIS models in Dispatch

We can add "LUIS Application" into Luis dispatch application (Auto generated application through the dispatch command tool and the version will be denoted as "Dispatch" instead of "1.0"). Here Luis dispatch application acts as a parent app because while running "*dispatch refresh*" command the LUIS app will push all the latest updates to Luis dispatch application (Parent). So, this will be easy for maintenance of multiple models in LUIS applications.

dispatch add -t luis -i "<app-id-for-luis-test-app>" -n "<name-of-luis-test-app>" -v <app-version-number> -k "<your-luis-authoring-key>" --intentName <intent-name>

We can get all the required command parameters for the LUIS app creation in dispatcher from the LUIS application that we have created. For this, click on the already created LUIS app and Go to "Settings" tab under the Manage menu and it will provide all the relevant information.



Application Settings

App name

LuisTestingApp

App description (optional)

The application using for testing

App ID

4068d4ee-f3b3-4f7c-9220-baeb983a6bd4

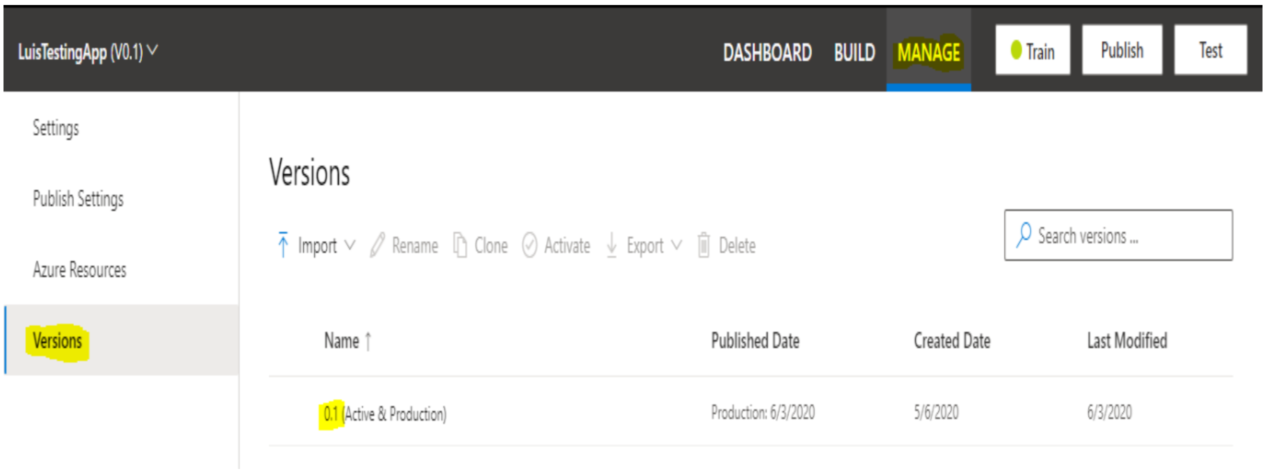
Culture

en-us

app-id-for-luis-test-app: This will be the ID of the LUIS application and we can copy from the application ID section in the above screenshot.

name-of-luis-test-app: Any name we can give as an application name but for the future maintenance better to give as the same application name.

app-version-number: The LUIS application version we can copy from under "Versions" tab under the Manage menu and it will provide all the relevant information or Left top of the above picture.



Name ↑	Published Date	Created Date	Last Modified
0.1 (Active & Production)	Production: 6/3/2020	5/6/2020	6/3/2020

Intent-name: As we mention the dispatch will find the intent based on the utterances that we are searching in the dispatch created luis application. The intent name should be required for adding any models in the LUIS application.

This command will update already created ". dispatch" file with LUIS model information and also remember all this information now added in the target machine ".dispatch" file and not in the LUIS application.

dispatch add -t luis -i "4068d4ee-f3b3-4f7c-9220-baeb983a6bd4" -n "LuisTestingApp" -v 0.1 -k "8f34728f544e401d8e5e7b89636fd5f9" --intentName l_cog_luis_test

The finally the luis application information added in the ".dispatch" file in the target machine and here is the updated ".dispatch" file.

```
{
  "authoringRegion": "westus",
  "hierarchical": true,
  "useAllTrainingData": false,
  "dontReviseUtterance": false,
  "copyLuisData": true,
"normalizeDiacritics": true,
"services": [
 {
   "intentName": "q_test-qna",
   "includePrompts": true,
   "kbId": "9b334bc4-af33-4cab-a7ff-e252b9cd5754",
   "subscriptionKey": "0f4399fce1c24c7197a294a88b28dcb4",
   "type": "qna",
   "name": "qna-cog-test",
   "id": "1"
 },
 {
   "intentName": "l_cog_luis_test",
   "appId": "4068d4ee-f3b3-4f7c-9220-baeb983a6bd4",
   "authoringKey": "8f34728f544e401d8e5e7b89636fd5f9",
   "version": "0.1",
   "region": "westus",
   "type": "luis",
   "name": "LuisTestingApp",
   "id": "2"
 }
],
"serviceIds": [
 "1",
 "2"
],
"authoringKey": "8f34728f544e401d8e5e7b89636fd5f9",
"version": "Dispatch",
  "region": "westus",
  "type": "dispatch",
  "name": "luis-tst"
}
```

Add another source to Dispatch model

We can add multiple sources to dispatch models into .dispatch files other than Cognitive Services QnA and LUIS models.

dispatch add -t file -n TsvModule -f c:\src\tsvmodule.tsv

dispatch add -t file -n TextModule -f c:\src\textmodule.txt

dispatch add -t file -n JsonModule -f c:\src\jsonmodule.json

dispatch add -t file -f c:\src\rajeesh\filepathmodule.tsv --intentName l_FilePathModule

The following are the supported file types.

- .tsv - Lines of tab delimited fields of intent and utterance (in that order).
- .txt - Lines of utterances with intent as file name.
- .json - Exported LUIS or QnA Maker json file.

Remove models from Dispatch application

There are a few situations which may occur in the future to remove any chatbot models from the dispatch luis model application, either QnA or LUIS, based on the requirement. The following command will remove any of the models from the dispatch application.

dispatch remove -t luis -i xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx

dispatch remove -t qna -i xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx

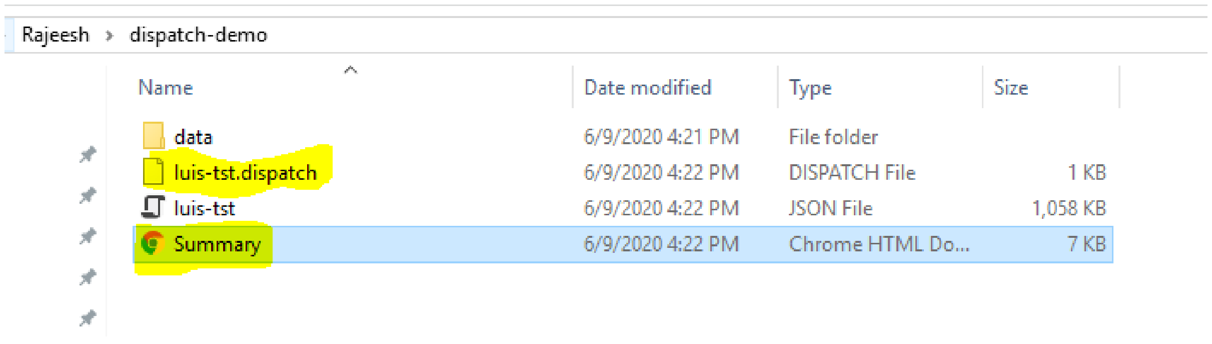
dispatch remove -t file -f c:\src\billmodule.json

Dispatch Luis App Creation Command

The following command will create, train and publish the Luis Dispatch Application with consists of multiple models that we have added in the dispatch application. This will be a one-time process and it will generate multiple files including the ".dispatch" file with all the relevant application information.

dispatch create

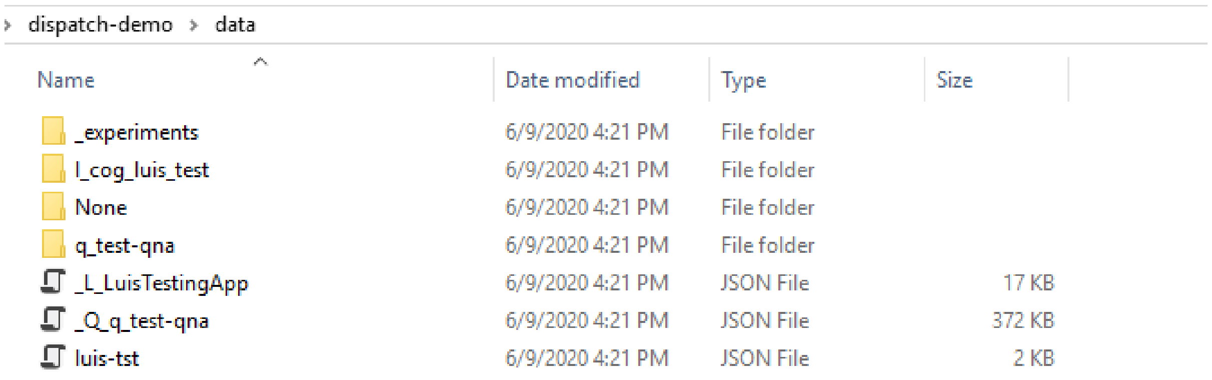
Once we've created the dispatch file then the folder structure will be created in the target machine for the respective directory and it will look like below:



Rajeesh › dispatch-demo

Name	Date modified	Type	Size
data	6/9/2020 4:21 PM	File folder	
luis-tst.dispatch	6/9/2020 4:22 PM	DISPATCH File	1 KB
luis-tst	6/9/2020 4:22 PM	JSON File	1,058 KB
Summary	6/9/2020 4:22 PM	Chrome HTML Do...	7 KB

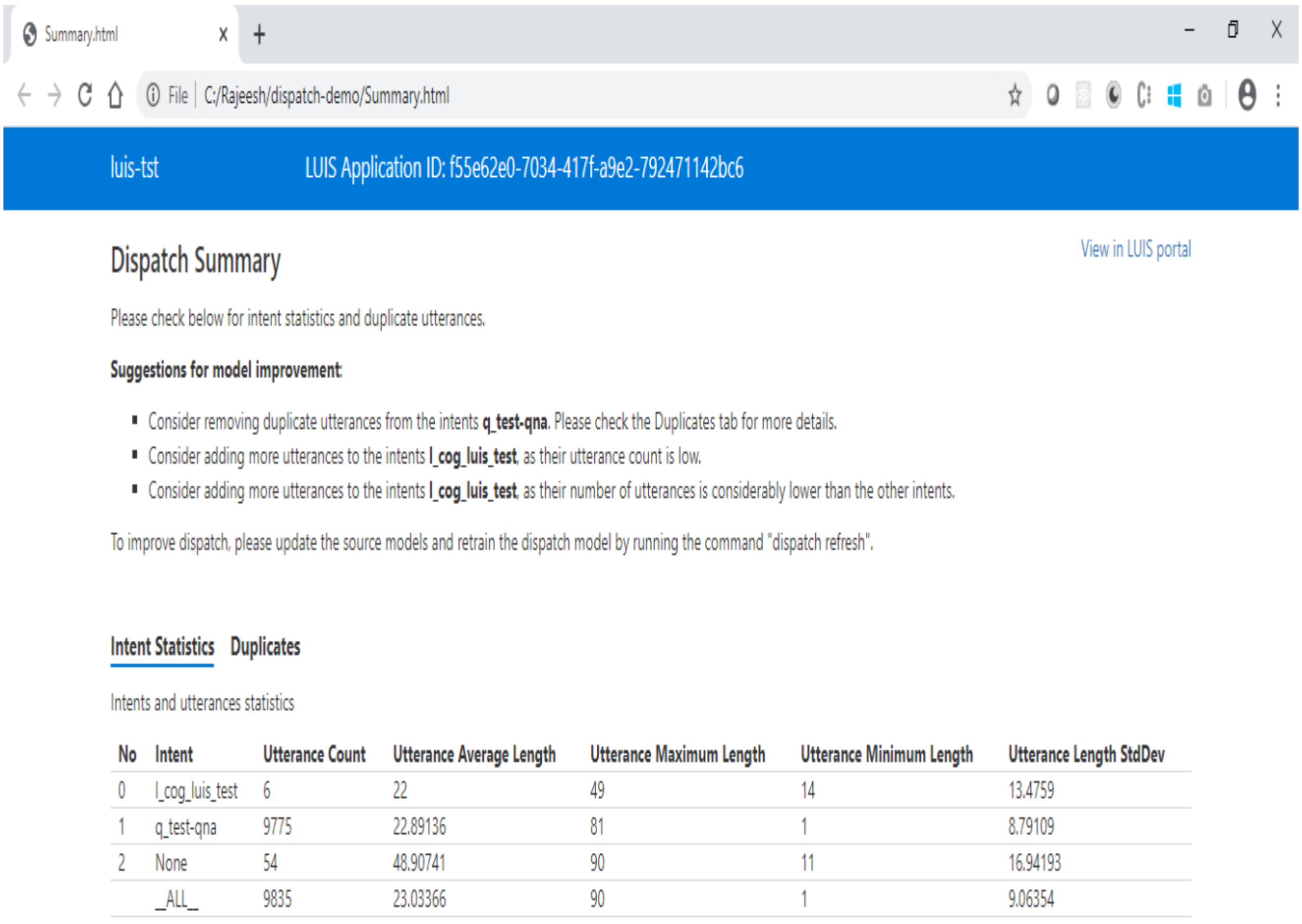
The data folder will contain each intent name and the utterances details as text file information.



dispatch-demo › data

Name	Date modified	Type	Size
_experiments	6/9/2020 4:21 PM	File folder	
l_cog_luis_test	6/9/2020 4:21 PM	File folder	
None	6/9/2020 4:21 PM	File folder	
q_test-qna	6/9/2020 4:21 PM	File folder	
_L_LuisTestingApp	6/9/2020 4:21 PM	JSON File	17 KB
_Q_q_test-qna	6/9/2020 4:21 PM	JSON File	372 KB
luis-tst	6/9/2020 4:21 PM	JSON File	2 KB

The summary html file will contain all the detailed information about the intents that we have used in dispatch Luis application.



Summary.html

File | C:/Rajeesh/dispatch-demo/Summary.html

luis-tst LUIS Application ID: f55e62e0-7034-417f-a9e2-792471142bc6

Dispatch Summary

View in LUIS portal

Please check below for intent statistics and duplicate utterances.

Suggestions for model improvement:

- Consider removing duplicate utterances from the intents **q_test-qna**. Please check the Duplicates tab for more details.
- Consider adding more utterances to the intents **l_cog_luis_test**, as their utterance count is low.
- Consider adding more utterances to the intents **l_cog_luis_test**, as their number of utterances is considerably lower than the other intents.

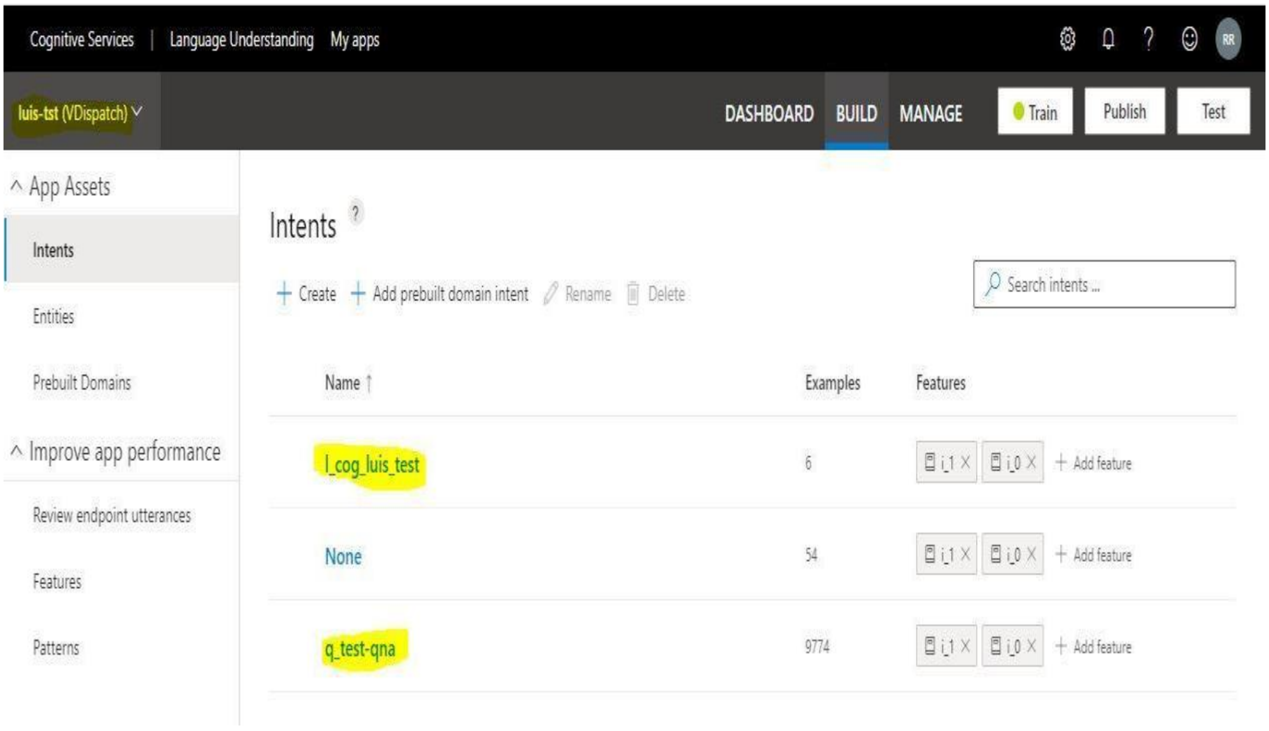
To improve dispatch, please update the source models and retrain the dispatch model by running the command "dispatch refresh".

Intent Statistics **Duplicates**

Intents and utterances statistics

No	Intent	Utterance Count	Utterance Average Length	Utterance Maximum Length	Utterance Minimum Length	Utterance Length StdDev
0	l_cog_luis_test	6	22	49	14	13.4759
1	q_test-qna	9775	22.89136	81	1	8.79109
2	None	54	48.90741	90	11	16.94193
	ALL	9835	23.03366	90	1	9.06354

The dispatch file created in the LUIS application has all the model's information like qna and luis application and it will contain the utterances, phrase list, entities, etc. Also, the version of the dispatch Luis application will be “VDispatch” instead of “1.0” because the “1.0” version denotes manually created luis application.



Name	Examples	Features
l_cog_luis_test	6	i_1 i_0 Add feature
None	54	i_1 i_0 Add feature
q_test-qna	9774	i_1 i_0 Add feature

We can cross verify the dispatch version of the Luis application name and key in the ".dispatch" file in the target machine. Here is the response of the dispatch luis application and it contains the dispatch luis application app id and name.

```
{
 "authoringRegion": "westus",
 "hierarchical": true,
 "useAllTrainingData": false,
 "dontReviseUtterance": false,
 "copyLuisData": true,
 "normalizeDiacritics": true,
 "services": [
  {
   "intentName": "q_test-qna",
   "includePrompts": true,
   "kbId": "9b334bc4-af33-4cab-a7ff-e252b9cd5754",
   "subscriptionKey": "0f4399fce1c24c7197a294a88b28dcb4",
   "type": "qna",
   "name": "qna-cog-test",
   "id": "1"
  },
  {
   "intentName": "l_cog_luis_test",
   "appId": "4068d4ee-f3b3-4f7c-9220-baeb983a6bd4",
   "authoringKey": "8f34728f544e401d8e5e7b89636fd5f9",
   "version": "0.1",
   "region": "westus",
   "type": "luis",
   "name": "LuisTestingApp",
   "id": "2"
  }
 ],
 "serviceIds": [
  "1",
  "2"
 ],
 "appId": "f55e62e0-7034-417f-a9e2-792471142bc6",
 "authoringKey": "8f34728f544e401d8e5e7b89636fd5f9",
 "version": "Dispatch",
 "region": "westus",
 "type": "dispatch",
 "name": "luis-tst"
}
```

Testing your dispatch model

To test dispatch model against test set:

dispatch test [options]

Evaluating your dispatch model

The following command will execute evaluation on the dispatch model to verify the cross validation and generate a summary of the evaluation:

dispatch eval [options]

Update Dispatch Luis Application

The *“dispatch refresh”* command will be published for the new changes in the already created dispatch application.

In the local machine we can use the following command.

dispatch refresh

In the remote machine we can use the following command.

dispatch refresh --remote true

Create the dispatch model

The Command line interface for the dispatch tool creates the model for dispatch version of dispatch including other models LUIS or QnA Maker app.

Open a command prompt or terminal window, and change directories to the CognitiveModels directory or any other folder that we are going to maintain for the Dispatch model because the command will be auto generated depending on the file for dispatch model.

Step 1:

The npm version should be 8.5 or above required for Dispatch tool installation and make sure you have the current version of npm and the Dispatch tool.

npm i -g npm

npm i -g botdispatch

Step 2:

The .dispatch file contains all the model information including authorization key. Use dispatch init to initialize and create a .dispatch file for your dispatch model.

dispatch init -n <filename-to-create> --luisAuthoringKey "<your-luis-authoring-key>" --luisAuthoringRegion <your-region>

Step 3:

Add LUIS apps and QnA Maker knowledge bases to the .dispatch file using add command for respective models.

dispatch add -t qna -i "<knowledge-base-id>" -n "<knowledge-base-name>" -k "<azure-qna-service-key1>" --intentName q_test-qna

dispatch add -t luis -i "<app-id-for-luis-test-app>" -n "<name-of-luis-test-app>" -v <app-version-number> -k "<your-luis-authoring-key>" --intentName l_luis-test

dispatch add -t file -n TsvModule -f c:\src\tsvmodule.tsv

dispatch add -t file -n TextModule -f c:\src\textmodule.txt

dispatch add -t file -n JsonModule -f c:\src\jsonmodule.json

dispatch add -t file -f c:\src\rajeesh\filepathmodule.tsv --intentName l_FilePathModule

Note: A maximum of 500 dispatch sources could be added to a Dispatch model.

Step 4:

Use dispatch create command to generate a dispatch model from the .dispatch file.

dispatch create

Publish the dispatch LUIS app you just created.

Pros and Cons of of Dispatch model

- Handle multiple models at a time.
- Reduce the time for training and publishing multiple models.
- Easy to use and very fast for identifying the intent and utterances details.
- Only support for Node package 8.5 and above.
- Manually added intent and utterances will be removed from the dispatch model.
- It never captures the entity details of the utterances.
- The LUIS application should contain unique utterances to include all intents never allow the duplicate entries.
- The duplicate utterances restriction in all intents will create an issue while having the same questions in multiple qna maker chatbot models. E.g : What is the product location?
- The direct entity mapping is restricted in the LUIS dispatch application and while pushing itself those entity mappings will be removed.
- The pattern is used for the entity mapping and we can include any of the types.
- The utterance limit as of now is 15k and this will be impacted when you handle a large application. For example, if you have multiple departments in an organization, each department contains unique questions approximately 30 thousand. In this scenario, luis dispatch can't load all the utterances or questions for respective department and it will skip out few utterances or questions for the performance and maintainability.

References

- https://docs.microsoft.com/en-us/azure/cognitive-services/luis/
- https://docs.microsoft.com/en-us/azure/cognitive-services/qnamaker/
- https://docs.microsoft.com/en-us/azure/guides/developer/azure-developer-guide
- https://docs.microsoft.com/en-us/azure/cognitive-services/what-are-cognitive-services
- https://docs.microsoft.com/en-us/azure/bot-service/bot-builder-tutorial-dispatch?view=azure-bot-service-4.0&tabs=cs