**Create intents and add utterances**

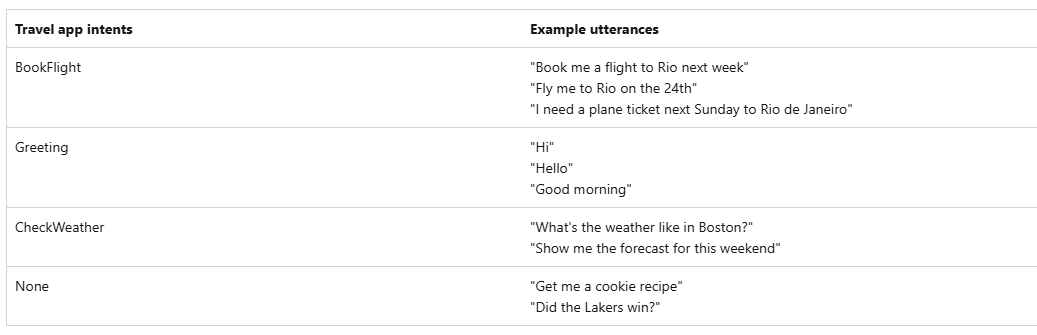
**Important**

LUIS will be retired on October 1st 2025 and starting April 1st 2023 you will not be able to create new LUIS resources. We recommend [migrating your LUIS applications](https://learn.microsoft.com/en-us/azure/ai-services/language-service/conversational-language-understanding/how-to/migrate-from-luis) to [conversational language understanding](https://learn.microsoft.com/en-us/azure/ai-services/language-service/conversational-language-understanding/overview) to benefit from continued product support and multilingual capabilities.

**Intents**

An intent represents a task or action the user wants to perform. It is a purpose or goal expressed in a user's [utterance](https://learn.microsoft.com/en-us/azure/ai-services/luis/concepts/utterances).

Define a set of intents that corresponds to actions users want to take in your application. For example, a travel app would have several intents:



All applications come with the predefined intent, "[None](https://learn.microsoft.com/en-us/azure/ai-services/luis/concepts/intents#none-intent)", which is the fallback intent.

**Prebuilt intents**

LUIS provides prebuilt intents and their utterances for each of its prebuilt domains. Intents can be added without adding the whole domain. Adding an intent is the process of adding an intent and its utterances to your app. Both the intent name and the utterance list can be modified.

**Return all intents' scores**

You assign an utterance to a single intent. When LUIS receives an utterance, by default it returns the top intent for that utterance.

If you want the scores for all intents for the utterance, you can provide a flag in the query string of the prediction API.

A screenshot of a computer

Description automatically generated

**Intent compared to entity**

The intent represents the action the application should take for the user, based on the entire utterance. An utterance can have only one top-scoring intent, but it can have many entities.

Create an intent when the user's intention would trigger an action in your client application, like a call to the checkweather() function from the table above. Then create entities to represent parameters required to execute the action.



**None intent**

The **None** intent is created but left empty on purpose. The **None** intent is a required intent and can't be deleted or renamed. Fill it with utterances that are outside of your domain.

The **None** intent is the fallback intent, and should have 10% of the total utterances. It is important in every app, because it’s used to teach LUIS utterances that are not important in the app domain (subject area). If you do not add any utterances for the **None** intent, LUIS forces an utterance that is outside the domain into one of the domain intents. This will skew the prediction scores by teaching LUIS the wrong intent for the utterance.

When an utterance is predicted as the None intent, the client application can ask more questions or provide a menu to direct the user to valid choices.

**Negative intentions**

If you want to determine negative and positive intentions, such as "I **want** a car" and "I **don't** want a car", you can create two intents (one positive, and one negative) and add appropriate utterances for each. Or you can create a single intent and mark the two different positive and negative terms as an entity.

**Intents and patterns**

If you have example utterances, which can be defined in part or whole as a regular expression, consider using the [regular expression entity](https://learn.microsoft.com/en-us/azure/ai-services/luis/concepts/entities#regex-entity) paired with a [pattern](https://learn.microsoft.com/en-us/azure/ai-services/luis/concepts/patterns-features).

Using a regular expression entity guarantees the data extraction so that the pattern is matched. The pattern matching guarantees an exact intent is returned.

**Intent balance**

The app domain intents should have a balance of utterances across each intent. For example, do not have most of your intents with 10 utterances and another intent with 500 utterances. This is not balanced. In this situation, you would want to review the intent with 500 utterances to see if many of the intents can be reorganized into a [pattern](https://learn.microsoft.com/en-us/azure/ai-services/luis/concepts/patterns-features).

The **None** intent is not included in the balance. That intent should contain 10% of the total utterances in the app.

**Best Practices for Intents:**

**Define distinct intents**

Make sure the vocabulary for each intent is just for that intent and not overlapping with a different intent. For example, if you want to have an app that handles travel arrangements such as airline flights and hotels, you can choose to have these subject areas as separate intents or the same intent with entities for specific data inside the utterance.

If the vocabulary between two intents is the same, combine the intent, and use entities.

Consider the following example utterances:

1. Book a flight
2. Book a hotel

"Book a flight" and "book a hotel" use the same vocabulary of "book a *<noun>*". This format is the same so it should be the same intent with the different words of flight and hotel as extracted entities.

**Do add features to intents**

Features describe concepts for an intent. A feature can be a phrase list of words that are significant to that intent or an entity that is significant to that intent.

**Do find sweet spot for intents**

Use prediction data from LUIS to determine if your intents are overlapping. Overlapping intents confuse LUIS. The result is that the top scoring intent is too close to another intent. Because LUIS does not use the exact same path through the data for training each time, an overlapping intent has a chance of being first or second in training. You want the utterance's score for each intention to be farther apart, so this variance doesn't happen. Good distinction for intents should result in the expected top intent every time.

**Balance utterances across intents**

For LUIS predictions to be accurate, the quantity of example utterances in each intent (except for the None intent), must be relatively equal.

If you have an intent with 500 example utterances and all your other intents with 10 example utterances, the 500-utterance intent will have a higher rate of prediction.

**Add example utterances to none intent**

This intent is the fallback intent, indicating everything outside your application. Add one example utterance to the None intent for every 10 example utterances in the rest of your LUIS app.

**Don't add many example utterances to intents**

After the app is published, only add utterances from active learning in the development lifecycle process. If utterances are too similar, add a pattern.

**Don't mix the definition of intents and entities**

Create an intent for any action your bot will take. Use entities as parameters that make that action possible.

For example, for a bot that will book airline flights, create a **BookFlight** intent. Do not create an intent for every airline or every destination. Use those pieces of data as [entities](https://learn.microsoft.com/en-us/azure/ai-services/luis/concepts/entities) and mark them in the example utterances.

**Utterances**

Utterances are inputs from users that your app needs to interpret. To train LUIS to extract intents and entities from these inputs, it's important to capture various different example utterances for each intent. Active learning, or the process of continuing to train on new utterances, is essential to the machine-learning intelligence that LUIS provides.

Collect utterances that you think users will enter. Include utterances, which mean the same thing but are constructed in various ways:

* Utterance length - short, medium, and long for your client-application
* Word and phrase length
* Word placement - entity at beginning, middle, and end of utterance
* Grammar
* Pluralization
* Stemming
* Noun and verb choice
* [Punctuation](https://learn.microsoft.com/en-us/azure/ai-services/luis/luis-reference-application-settings#punctuation-normalization) - using both correct and incorrect grammar

**Utterances aren't always well formed**

Your app might need to process sentences, like "Book a ticket to Paris for me," or a fragment of a sentence, like "Booking" or "Paris flight" Users also often make spelling mistakes. When planning your app, consider whether or not you want to use [Bing Spell Check](https://learn.microsoft.com/en-us/azure/ai-services/luis/luis-tutorial-bing-spellcheck) to correct user input before passing it to LUIS.

If you don't spell check user utterances, you should train LUIS on utterances that include typos and misspellings.

**Use the representative language of the user**

When choosing utterances, be aware that what you think are common terms or phrases might not be common for the typical user of your client application. They might not have domain experience or use different terminology. Be careful when using terms or phrases that a user would only say if they were an expert.

**Choose varied terminology and phrasing**

You'll find that even if you make efforts to create varied sentence patterns, you'll still repeat some vocabulary. For example, the following utterances have similar meaning, but different terminology and phrasing:

* "*How do I get a computer?*"
* "*Where do I get a computer?*"
* "*I want to get a computer, how do I go about it?*"
* "*When can I have a computer?*"

The core term here, *computer*, isn't varied. Use alternatives such as desktop computer, laptop, workstation, or even just machine. LUIS can intelligently infer synonyms from context, but when you create utterances for training, it's always better to vary them.

**Example utterances in each intent**

Each intent needs to have example utterances - at least 15. If you have an intent that doesn't have any example utterances, you will not be able to train LUIS. If you have an intent with one or few example utterances, LUIS might not accurately predict the intent.

**Add small groups of utterances**

Each time you iterate on your model to improve it, don't add large quantities of utterances. Consider adding utterances in quantities of 15. Then [Train](https://learn.microsoft.com/en-us/azure/ai-services/luis/how-to/train-test), [publish](https://learn.microsoft.com/en-us/azure/ai-services/luis/how-to/publish), and [test](https://learn.microsoft.com/en-us/azure/ai-services/luis/how-to/train-test) again.

LUIS builds effective models with utterances that are carefully selected by the LUIS model author. Adding too many utterances isn't valuable because it introduces confusion.

It's better to start with a few utterances, then [review the endpoint utterances](https://learn.microsoft.com/en-us/azure/ai-services/luis/how-to/improve-application) for correct intent prediction and entity extraction.

**Utterance normalization**

Utterance normalization is the process of ignoring the effects of types of text, such as punctuation and diacritics, during training and prediction.

Utterance normalization settings are turned off by default. These settings include:

* Word forms
* Diacritics
* Punctuation

If you turn on a normalization setting, scores in the **Test** pane, batch tests, and endpoint queries will change for all utterances for that normalization setting.

When you clone a version in the LUIS portal, the version settings are kept in the new cloned version.

Set your app's version settings using the LUIS portal by selecting **Manage** from the top navigation menu, in the **Application Settings** page. You can also use the [Update Version Settings API](https://learn.microsoft.com/en-us/rest/api/luis/settings/update). See the [Reference](https://learn.microsoft.com/en-us/azure/ai-services/luis/luis-reference-application-settings) documentation for more information.

**Word forms**

Normalizing **word forms** ignores the differences in words that expand beyond the root.

**Diacritics**

Diacritics are marks or signs within the text, such as:

İ ı Ş Ğ ş ğ ö ü

**Punctuation marks**

Normalizing **punctuation** means that before your models get trained and before your endpoint queries get predicted, punctuation will be removed from the utterances.

Punctuation is a separate token in LUIS. An utterance that contains a period at the end is a separate utterance than one that doesn't contain a period at the end, and might get two different predictions.

If punctuation isn't normalized, LUIS doesn't ignore punctuation marks by default because some client applications might place significance on these marks. Make sure to include example utterances that use punctuation, and ones that don't, for both styles to return the same relative scores.

Make sure the model handles punctuation either in the example utterances (both having and not having punctuation) or in [patterns](https://learn.microsoft.com/en-us/azure/ai-services/luis/concepts/patterns-features) where it is easier to ignore punctuation. For example: I am applying for the {Job} position[.]

If punctuation has no specific meaning in your client application, consider [ignoring punctuation](https://learn.microsoft.com/en-us/azure/ai-services/luis/concepts/utterances#utterance-normalization) by normalizing punctuation.

**Ignoring words and punctuation**

If you want to ignore specific words or punctuation in patterns, use a [pattern](https://learn.microsoft.com/en-us/azure/ai-services/luis/concepts/patterns-features) with the *ignore* syntax of square brackets, [].

**Training with all utterances**

Training is nondeterministic: utterance prediction can vary slightly across versions or apps. You can remove nondeterministic training by updating the [version settings](https://learn.microsoft.com/en-us/rest/api/luis/settings/update) API with the UseAllTrainingData name/value pair to use all training data.

**Testing utterances**

Developers should start testing their LUIS application with real data by sending utterances to the [prediction endpoint](https://learn.microsoft.com/en-us/azure/ai-services/luis/luis-how-to-azure-subscription) URL. These utterances are used to improve the performance of the intents and entities with [Review utterances](https://learn.microsoft.com/en-us/azure/ai-services/luis/how-to/improve-application). Tests submitted using the testing pane in the LUIS portal aren't sent through the endpoint, and don't contribute to active learning.

**Review utterances**

After your model is trained, published, and receiving [endpoint](https://learn.microsoft.com/en-us/azure/ai-services/luis/luis-glossary#endpoint) queries, [review the utterances](https://learn.microsoft.com/en-us/azure/ai-services/luis/how-to/improve-application) suggested by LUIS. LUIS selects endpoint utterances that have low scores for either the intent or entity.

**Best practices**

**Label for word meaning**

If the word choice or word arrangement is the same, but doesn't mean the same thing, don't label it with the entity.

In the following utterances, the word fair is a homograph, which means it's spelled the same but has a different meaning:

* "*What kinds of county fairs are happening in the Seattle area this summer?*"
* "*Is the current 2-star rating for the restaurant fair?*

If you want an event entity to find all event data, label the word fair in the first utterance, but not in the second.

**Don't ignore possible utterance variations**

LUIS expects variations in an intent's utterances. The utterances can vary while having the same overall meaning. Variations can include utterance length, word choice, and word placement.

A white rectangular object with black lines

Description automatically generated

The second column uses different verbs (buy, reserve, book), different quantities (1, &"two", 3), and different arrangements of words but all have the same intention of purchasing airline tickets for travel.

**Don't add too many example utterances to intents**

After the app is published, only add utterances from active learning in the development lifecycle process. If utterances are too similar, add a pattern.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A computer screen with a computer screen

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated