**Create entities**

An entity is an item or an element that is relevant to the user's intent. Entities define data that can be extracted from the utterance and is essential to complete a user's required action. For example:

A close up of a paper

Description automatically generated

Entities are optional but recommended. You don't need to create entities for every concept in your app, only when:

* The client application needs the data, or
* The entity acts as a hint or signal to another entity or intent.

**Entity types**

To create an entity, you have to give it a name and a type. There are several types of entities in LUIS.

**List entity**

A list entity represents a fixed, closed set of related words along with their synonyms. You can use list entities to recognize multiple synonyms or variations and extract a normalized output for them. Use the *recommend* option to see suggestions for new words based on the current list.

A list entity isn't machine-learned, meaning that LUIS doesn’t discover more values for list entities. LUIS marks any match to an item in any list as an entity in the response.

Matching list entities is both case sensitive and it has to be an exact match. Normalized values are also used when matching the list entity. For example:

A white rectangular object with black lines

Description automatically generated

**Regex entity**

A regular expression entity extracts an entity based on a regular expression pattern you provide. It ignores case and ignores cultural variant. Regular expression entities are best for structured text or a predefined sequence of alphanumeric values that are expected in a certain format. For example:



**Prebuilt entities**

LUIS includes a set of prebuilt entities for recognizing common types of information, like dates, times, numbers, measurements, and currency. Prebuilt entity support varies by the culture of your LUIS app. For a full list of the prebuilt entities that LUIS supports, including support by culture, see the [prebuilt entity reference](https://learn.microsoft.com/en-us/azure/ai-services/luis/luis-reference-prebuilt-entities).

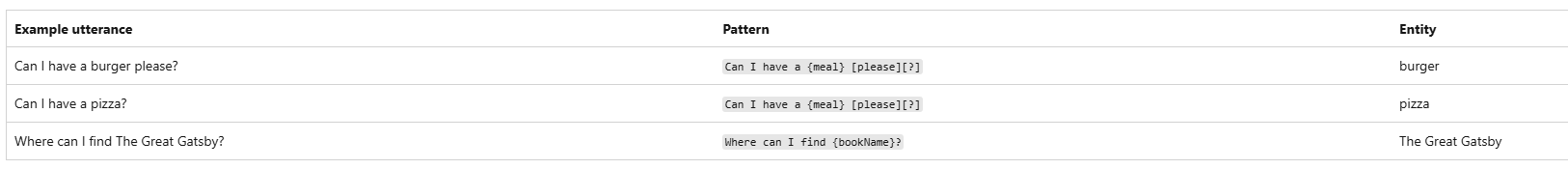
When a prebuilt entity is included in your application, its predictions are included in your published application. The behavior of prebuilt entities is pre-trained and can’t be modified.

A screenshot of a phone

Description automatically generated

**Pattern.Any entity**

A pattern.Any entity is a variable-length placeholder used only in a pattern's template utterance to mark where the entity begins and ends. It follows a specific rule or pattern and best used for sentences with fixed lexical structure. For example:



**Machine learned (ML) entity**

Machine learned entity uses context to extract entities based on labeled examples. It is the preferred entity for building LUIS applications. It relies on machine-learning algorithms and requires labeling to be tailored to your application successfully. Use an ML entity to identify data that isn’t always well formatted but have the same meaning.

A white rectangular object with a black border

Description automatically generated with medium confidence

**ML Entity with Structure**

An ML entity can be composed of smaller sub-entities, each of which can have its own properties. For example, an *Address* entity could have the following structure:

* Address: 4567 Main Street, NY, 98052, USA
  + Building Number: 4567
  + Street Name: Main Street
  + State: NY
  + Zip Code: 98052
  + Country: USA

**Building effective ML entities**

To build machine learned entities effectively, follow these best practices:

* If you have a machine learned entity with sub-entities, make sure that the different orders and variants of the entity and sub-entities are presented in the labeled utterances. Labeled example utterances should include all valid forms, and include entities that appear and are absent and also reordered within the utterance.
* Avoid overfitting the entities to a fixed set. Overfitting happens when the model doesn't generalize well, and is a common problem in machine-learning models. This implies the app wouldn’t work on new types of examples adequately. In turn, you should vary the labeled example utterances so the app can generalize beyond the limited examples you provide.
* Your labeling should be consistent across the intents. This includes even utterances you provide in the *None* intent that includes this entity. Otherwise the model won’t be able to determine the sequences effectively.

**Entities as features**

Another important function of entities is to use them as features or distinguishing traits for another intents or entities so that your system observes and learns through them.

**Entities as features for intents**

You can use entities as a signal for an intent. For example, the presence of a certain entity in the utterance can distinguish which intent does it fall under.



**Entities as Feature for entities**

You can also use entities as an indicator of the presence of other entities. A common example of this is using a prebuilt entity as a feature for another ML entity. If you’re building a flight booking system and your utterance looks like 'Book me a flight from Cairo to Seattle', you likely will have *Origin City* and *Destination City* as ML entities. A good practice would be to use the prebuilt GeographyV2 entity as a feature for both entities.

You can also use entities as required features for other entities. This helps in the resolution of extracted entities. For example, if you’re creating a pizza-ordering application and you have a Size ML entity, you can create SizeList list entity and use it as a required feature for the Size entity. Your application will return the normalized value as the extracted entity from the utterance.

**Data from entities**

Most chat bots and applications need more than the intent name. This additional, optional data comes from entities discovered in the utterance. Each type of entity returns different information about the match.

A single word or phrase in an utterance can match more than one entity. In that case, each matching entity is returned with its score.

All entities are returned in the entities array of the response from the endpoint

**Best practices for entities**

**Use machine-learning entities**

Machine learned entities are tailored to your app and require labeling to be successful. If you aren’t using machine learned entities, you might be using the wrong entities.

Machine learned entities can use other entities as features. These other entities can be custom entities such as regular expression entities or list entities, or you can use prebuilt entities as features.

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 computer screen shot of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Use ChatGPT to create sample utterances

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated