



Defining Intents and Entities



In the previous module, as we walked through the life of a conversation, we highlighted the elements that users have come to expect from today's conversational agents... like being able to understand user intention, having awareness of context, etc.

In this module, you will learn how to create a conversational agent in Dialogflow, and use Dialogflow's console to program the intents, provide training phrases and specify entities to train your agent to have these capabilities. To explain these concepts, we will use the example of a pizza ordering chatbot and show you what the intents and entities may look like. We will also share some best practices along the way. So let's dive into it.....

Agenda

Defining Intents for Your Agent

Identifying Entities in Your Intent



Let's start by defining intents for your agent.

It all starts with intents

Intents are the actions your users want to execute.

Intents represent something a user might ask for.



It all starts with intents.

Intents are the trunk and joints of the dialog tree. It connects all of the branches. Intents determine where a conversation will go and what an agent should do.

In communication, intents can be thought of as the root verbs in the dialog such as wanting coffee translating to acquiring a beverage. Sometimes the intents are not explicit and instead are inferred from the entire composition of a phrase.

You want to map your intents to the goal of your application. If you are a help desk application then your intents might include: opening/filing a ticket, updating a ticket, closing a ticket. But your application may also need to access and update a user's account information, branch over to a live technician, and even pass along a quality assurance survey. Even affirmation, answering yes or no, is an intent.

Intents evolve as your understanding of the users needs evolves and you may find yourself doubling or tripling your intents beyond the draft set of intents.

Intents help the agent determine what users want

2 mocha coffees please
Intent: Order Coffee

How many calories in a slice of pizza?
Intent: Get Nutritional Details

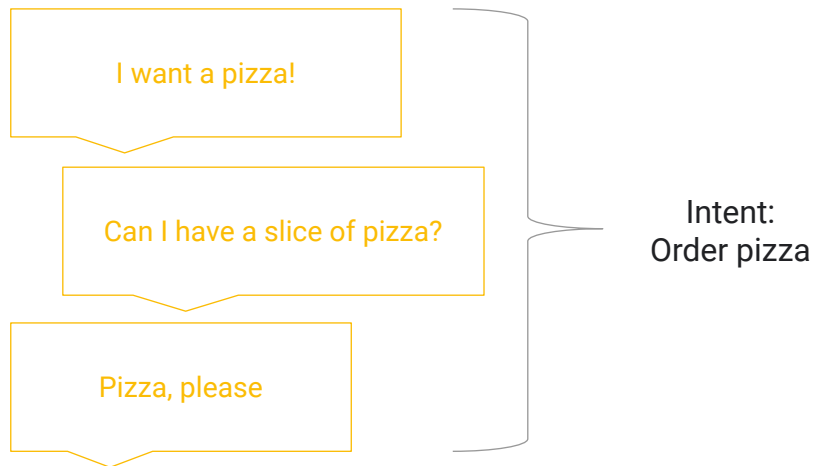
I want a pizza!
Intent: Order Pizza



To make the task of defining intents easier, some rules of thumb can be applied. First, identify the "verbs" in the dialog people will have with your agent. Doing that will allow your agent to have its actions mapped to needs from the user. Another possible scenario is to identify where the application should branch logic.

Here are some examples: *2 mocha coffees please*, *How many calories in a slice?*, *I want a pizza!* or even: *I really need some caffeine.*

Choosing the training phrases



Once you have chosen your intents, you need to "train" your agent to recognize them. This can be done with the use of training phrases.

The training phrases for each intent should be representative of how users manifest such intent. It is always a good idea to add variations to the grammatical construction of a request (passive vs active voices, questions vs statements, etc).

When creating an intent, the more training phrases you can think of, the better. Later we will discuss best practices for writing your training phrases.

Activity

Identifying Training Phases

Can you come up with at least 10 different ways of ordering pizza?

Your training phrases can be generic or specific.



Let's do a quick activity.....

Can you come up with at least 10 different ways of ordering pizza? Your training phrases can be generic (I want a pizza!) or specific (Can I have a slice of pepperoni pizza?). Needless to say, you could even add details such as the type of crust. It all depends on your use case!

Take a few seconds to think about it

Some training phrases for *Order Pizza* intent

Can I have a slice of pizza please?

I want a pizza now

Whole pizza with peppers and bacon

I'd like to order pizza

One slice of veggie pizza

Get me a pineapple and ham pizza

Is it possible to buy a pizza?

I want a pizza with mushroom, bacon, and cheese



Here are some examples of additional training phrases. I will use these examples when I create my pizza ordering intent in the Dialogflow console. Let's see how we can do that.

Demo

Creating Intents

In this demo, we'll create a pizza ordering intent in the Dialogflow console.



Switch to demo where we see these same examples, then we try the example below.
(Agent: agent-1)

Alright....I am in the the Dialogflow console (which is at console.dialogflow.com). As you can see, the left side bar contains a menu with many options. As you can see, I've already gone ahead and created my agent, called "agent-1" and now I am ready to set up my intents.

On the intents page we can create new intents. By clicking Create Intent, you can define the order pizza we have discussed in the previous videos. Choose a name for the intent. Then scroll down to the training phrases section to add the training phrases you came up with.

Once you finished, move down to the response section. What do you want your agent to reply to the user once it detects that the user wants a pizza? If you want the agent to reply with a sentence, simply add the sentence to the response section. Otherwise, if the detected intent requires an action to be taken instead, you can leave the response section blank. Click save once you are done. You will notice that the agent automatically starts training once changes are introduced. Wait until the training has completed so you can test your changes.

At this point, I am going to switch to another agent that I had already setup previously, with more training phrases.

Complete agent from demo is agent-2

You can test the agent by typing a request on the right side bar.

Possible example:

- Try: can I pick up a cheese pizza in 2 hours?

Notice that even though the example entered is not part of the training phrases, the agent was still able to map it to the right intent.

Also, besides the intents that I just created, you may have noticed 2 other intents that were already there to begin with...the default welcome and fallback intent. Default welcome intent handles all types of greetings like the hi, hello etc. so we dont need to worry about catching those. As for fallback intent...well lets talk more about it.

Other types of intents

1 Fallback intent

2 Follow-up intent



There is a special type of intent called Fallback intent. Fallback intents are, as the name suggests, a fallback for when the agent does not understand what the user is asking for. You can try asking a question to your order agent about the weather and see what happens. You will notice that you don't need to specify the answers the agent will return for a fallback intent, and that is why this is an intent that is automatically created for each new agent.

There is also an intent called follow-up intent. We will talk about this when we cover context in the next module.

Best practices when defining intents



Training phrases should be representative.



Make sure your intents do not overlap.



Prioritize intents that represent the most common requests.



Here are some best practices to follow when defining intents for your agent. When choosing the training phrases to train an intent, make sure to consider the different ways users might convey that intention. This can vary from diverse vocabulary to different grammatical constructions of a phrase or sentence. Another important aspect is to keep your intents non-ambiguous. That means avoiding situations where similar requests are mapped to different intents within the agent. It is always a good idea to start the development of your agent focused on addressing the most common requests from the users. This will allow the application to quickly take advantage of the ability of the agent to address users' requests.

Agenda

Defining Intents for Your Agent

Identifying Entities in Your Intent



So far you have learned about creating intents and using training phrases to teach the agent how to recognize these intents. But, let's say that you want to enable your agent to extract specific pieces of information provided by the user. For example, the toppings they want on the pizza they are ordering, or the number of slices.

You can do that with entities.

Use entities to extract useful facts from what users say

Entities help identify the:

Who

What

When

Where

in the natural language input.



Entities help you get to the specifics of an interaction. In dialog, the entities are the nouns or quantifiers found through the conversation such as a person's name, the food in a review of a cruise, specific numbers, dates, just to name a few.

As a developer, you can easily define your own entities in the Dialogflow UI or programmatically

I'd like a **strawberry milkshake**
made with **2% milk**

Get me a **veggie** pizza!

How much do **mints** cost?



In the case of ordering pizza, pizza would be a grouping of attributes that can be seen as entities. Pizza ordering have as attributes: number of pizzas, toppings, type of crust, pickup or delivery time. When you are designing your entity groupings you need to know how granular the entities should get. Sometimes simply pizza is sufficient, but sometimes you need to know finer details. Entities help your agent decide what details it needs to know and how it should act based on those details.

Entities are also a great way to add personalization to an agent. You can use an entity and data stored in a database to remember details about a user such as their name or favorite order. You can then echo those details back turning a rigid conversation into a casual dialog.

Entities usually are the nouns in your dialog. They are commonly composed of root terms and their synonyms. For the example in the orange box, someone could say veggie or vegetarian and these should map to the same entity representing the toppings of the pizza. Let's see how we can create entities in the Dialogflow console.

Demo

Creating an Entity

In this demo, we'll create a new simple entity called `pizza_topping`, which will hold the values for the pizza toppings we carry.



Demo for showing how to create an entity: Simple entity, synonym automated extension (Agent: agent-3)

Script: So I am back in the Dialogflow console...and To create a new entity in Dialogflow, click on the Entities page. Then click Create Entity. Choose a name for your entity. For this example, we will create an entity called `pizza_topping`, which will hold the values for the pizza toppings we carry. You will see two options: Define synonyms (which is checked by default) and Allow automated expansion. Let's start with the Define synonyms option. Click to edit entry then type cheese for our first reference value. Hit tab. You will see that the word used as the reference value is automatically added to the list of synonyms. I won't add any synonyms for cheese because people usually refer to a cheese topping by using the cheese word. But if I add vegetarian, I might want to add veggie as one of the synonyms.

Completed agent for demo is agent-4

I am now gonna switch to another agent i had prepared that has a larger list of possible pizza toppings. One example where we may want to enable the option "Allow automated expansion" is when we want to allow the agent to accept other reference values that may be said by the user but were not added to the list initially. Let's say a user wants onions on their pizza. In this examples the only way our system will be able to add onions to the order is if the Allow automated expansion option is checked.

If you have a fixed set of options and would NOT want to allow new ones to be added, then you should not check this option. Click save.


Now we are ready to annotate our intent training phrases with entities. To do that, let's go back to our order.pizza intent. To annotate an entity on a training phrase, double click on the word you want to annotate. Let's annotate the phrase: I want a pizza with mushroom, bacon, and cheese. Double click on mushroom and choose @pizza_topping. Do the same with the bacon and cheese. Remember we need to be consistent during this phase so we don't annotate "peppers and bacon" as one entity, for example.

Again, I am gonna switch to another agent that has all training phrases annotated.

Completed agent for demo is agent-5


Let me show you what happens if you add a new training phrase. Let's add the phrase: I would like to order a beef, sausage and pepperoni pizza. Notice that the agent automatically annotates words that match entity reference values. You might be asking, then why don't we create entities before adding training phrases. And that is totally reasonable. And you will get to try that out in the hands on labs.


Mapping entity attributes through composite entities



drink

SAVE



☐ Define synonyms 

☐ Allow automated expansion

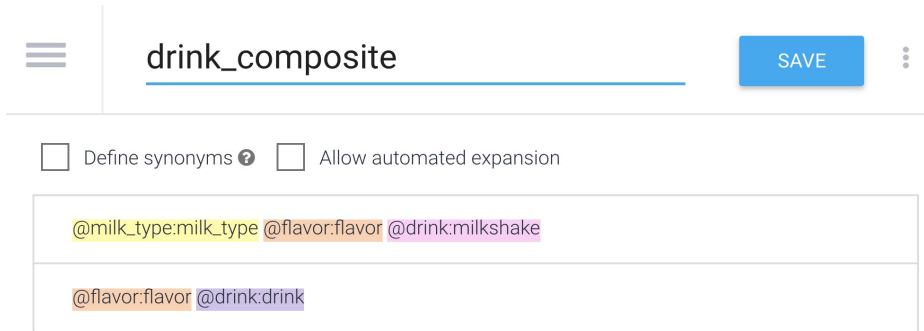
smoothie
milkshake
Enter value



When you are creating an entity and you identify that the entity contains attributes that you need to map to, one way is to use composite entities.

For our ordering example, let's say that we want to create an intent for ordering drinks. A drink can be of type milkshake or smoothie. Milkshake and smoothie, in this case, are entries in an entity called drink.

Creating complex entities with composite entities



The screenshot shows a configuration interface for a composite entity named "drink_composite". On the left is a hamburger menu icon. The entity name "drink_composite" is in a text field with a blue underline. To the right is a blue "SAVE" button and a vertical ellipsis menu. Below the entity name are two checkboxes: "Define synonyms" (unchecked) and "Allow automated expansion" (unchecked). There are two text input fields containing entity references. The first field contains "@milk_type:milk_type @flavor:flavor @drink:milkshake" with each reference highlighted in a different color (yellow, orange, and purple). The second field contains "@flavor:flavor @drink:drink" with each reference highlighted in orange and purple.

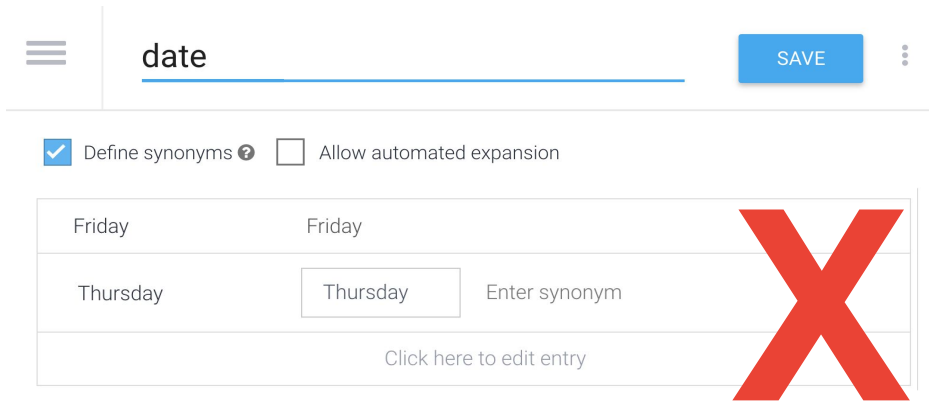


Additionally, let's say we have different flavors and milk types that can be chosen for the drinks. In this scenario, we might want to use composite entities to allow the agent to identify these attributes when the user requests: "can I have a non-fat strawberry milkshake?".

For this case, create separate entities for milk type (listing all of the types of milk your business carries), and a separate entity for flavor. Then, combine those under an entity (here we call it drink_composite) that shows the different ways customers can ask for the drink and its attributes.

This part should be one the lab

Don't create entities for dates



The screenshot shows a configuration interface for a date entity. At the top, there is a search bar with the word "date" entered, a "SAVE" button, and a menu icon. Below the search bar, there are two checkboxes: "Define synonyms" (checked) and "Allow automated expansion" (unchecked). A table below shows a list of synonyms for "date". The table has two columns: the first column contains the word "Friday" and the second column contains the word "Friday". The third row shows "Thursday" in the first column, "Thursday" in the second column, and "Enter synonym" in the third column. A link "Click here to edit entry" is at the bottom of the table. A large red "X" is overlaid on the right side of the table, indicating that this interface should not be used to create entities for dates.

Friday	Friday	
Thursday	Thursday	Enter synonym
Click here to edit entry		

What if you need to capture the date or time of an order?

For the pizza order use case, let's say that the user wants to determine the time they want to pick up their order. In that case, when they say the time, the agent should be able to identify and extract the time as a standard format (ISO, for example), so it can communicate the time to the backend system responsible for the orders.

Similarly, sometimes we need to identify common concepts such as dates, addresses, phone numbers, given names. For that, instead of defining entities for things such as days of the week, or months of the year

Use system entities instead

Entity Name	Description	Examples	Returned Data Type	Returned Object Example
@sys.date-time	Matches date, time, intervals or date and time together	2:30 pm 13 July April morning tomorrow at 4:30 pm tomorrow afternoon	String in ISO-8601 format or object: Strings in ISO-8601 format	"2018-04-05T14:30:00-06:00" "2018-07-13T18:00:00-06:00" { "startDate": "2018-04-01T12:00:00-06:00", "endDate": "2018-04-30T12:00:00-06:00" } { "endTime": "2018-04-06T12:00:00-06:00", "startTime": "2018-04-06T08:00:00-06:00" } { "date_time": "2018-04-06T16:30:00-06:00" } { "startDateTime": "2018-04-06T12:00:00-06:00", "endDateTime": "2018-04-06T16:00:00-06:00" }
@sys.date	Matches a date	tomorrow	String in ISO-8601 format	"2018-04-06T12:00:00-06:00"
@sys.date-period	Matches a date interval	April	"object: Strings in ISO-8601 format"	{ "startDate": "2018-04-01T12:00:00-06:00", "endDate": "2018-04-30T12:00:00-06:00" }
@sys.time	Matches a time	4:30 pm	String in ISO-8601 format	"2018-04-05T16:30:00-06:00"
@sys.time-period	Matches a time interval	afternoon	"object: Strings in ISO-8601 format"	{ "startTime": "2018-04-05T12:00:00-06:00", "endTime": "2018-04-05T16:00:00-06:00" }



Use one of the system entities that represent date and time.

System entities are pre-built entities provided by Dialogflow in order to facilitate handling the most popular concepts such as addresses, currency, units, date, time and many others.

Do you see any system entities that could be used in our pizza ordering intent? Let's check it out.

Demo

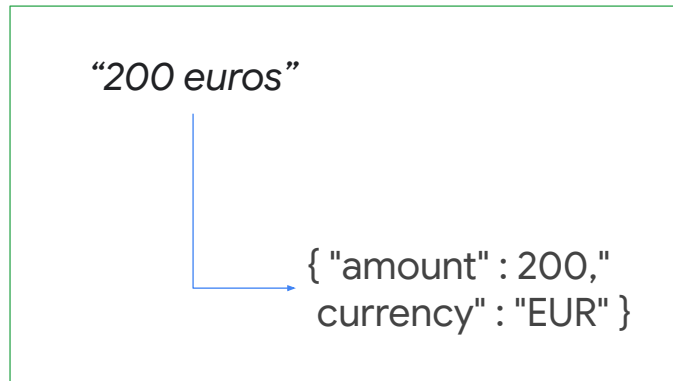
How to create an entity



Demo for showing how to create an entity: Simple entity, synonym automated extension

In the training phrases we have so far, we can annotate the number of pizzas "One" and the time the user wants the pizza to be prepared for pick up "now". Notice that system entity names are preceded by sys. To differentiate them from the developer entities.

System composite entity

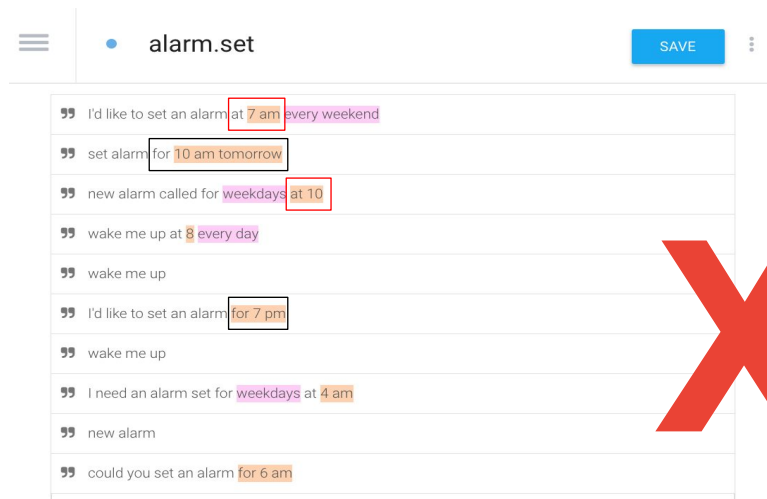


Up to this point we've looked at composite entities...and system entities,,,,individually. But what about system composite entities?

For example..If the user says 200 euros, Dialogflow returns an object type value consisting of two attribute-value pairs: amount and currency.

Visit the Reference section at the end of this module to find the documentation of all of the system entities currently supported.

Don't annotate inconsistently



The screenshot shows a training interface for an agent named "alarm.set". It features a list of training phrases with various parts highlighted by colored boxes to represent entities. The phrases and their annotations are:

- "I'd like to set an alarm at 7 am every weekend" (7 am: orange box, every weekend: pink box)
- "set alarm for 10 am tomorrow" (10 am: orange box, tomorrow: orange box)
- "new alarm called for weekdays at 10" (weekdays: pink box, at 10: orange box)
- "wake me up at 8 every day" (8: orange box, every day: pink box)
- "wake me up" (no annotations)
- "I'd like to set an alarm for 7 pm" (7 pm: orange box)
- "wake me up" (no annotations)
- "I need an alarm set for weekdays at 4 am" (weekdays: pink box, at 4 am: orange box)
- "new alarm" (no annotations)
- "could you set an alarm for 6 am" (for 6 am: orange box)

A large red "X" is overlaid on the right side of the list, indicating that this inconsistent annotation scheme is incorrect. A "SAVE" button is visible at the top right of the interface.



When creating entities, there's a few things to keep in mind to make sure we are training our agent well. Let's look at some best practices when creating entities.

First, it's important to be consistent when annotating the entities in the training phrases. This will help the agent to not get confused about what should be recognized as a given entity.

Note that in some cases they have included "at" and "for". Being careful to not add stop words such as prepositions is extremely important.

Don't forget to diversify examples

- horoscope

Contexts

User says

” Add user expression

” any new horoscopes for Taurus

” horoscope Taurus

” give me horoscope for Taurus

” check for Taurus

” horoscope for Taurus



Make sure you provide diversification of examples of a given entity in your training phrases. This allows the agent to understand that the concept is part of an intent, as opposed to an instance of the concept alone. On this example, providing only Taurus will not be enough to teach the agent that the same should apply to other constellations as well.

Summary

- 1 What are intents and how to train your agent to detect them
- 2 Fallback intents
- 3 What are entities and how to annotate them in your agent
- 4 Different types of entities
- 5 Some best practices



Alright...now let's quickly review the topics covered so far.

In this module you learned about what intents are and how to train your agent to detect them.

You also learned what fallback intents are and when they are triggered.

You learned about entities and how to go about annotating them in your training phrases so your agent can extract specific details from a conversation.

We saw that there are system entities, composite entities, and system composite entities.

And last but not the least, you saw some best practices when working with intents and entities.

In the next module, we will continue developing our pizza ordering agent and talk about how to maintain context. We will also write some code to write pizza orders to a database.

Group Activity

Design your own bot



Scenario analysis

- a. List all the possible intents
i.e. All the conversations regarding to pizza ordering should be considered. order pizza, delivery location, order drink
- b. Provide at least 10 sentences for each intent(Just do it for one intent)
- c. Identify Entities
i.e. Pizza toppings, drinks, etc.
- d. Consider fallback intent
- e. Include basic conversation(small talk)
i.e. hello, bye, ending conversation, ...

Lab

Lab 1,Part 1



Now that you are familiar with the concepts, let's work on the lab and get hands-on. You will notice that the lab will walk you through the creation of entities before you provide examples for training the intents. This is intentional because it allows the entities to be annotated automatically as you enter training phrases, so you don't have to. However, make sure the entities get recognized and annotated properly. If not, you can manually fix the annotations yourself.

cloud.google.com

