Building Chatbots with Zero Experience

Lex vs Dialogflow vs Watson vs Rasa

Contents

Which Chatbot Platforms are Easiest to Build and Train On? ... 1

Defining the Chatbot's Goal and Use Case ... 3

- Mapping out a rough conversation tree ... 3
- Finding out which chatbot is easiest to train ... 5

Training Google Dialogflow ... 8

- Documentation ... 8
- Configuration ... 8
- Testing ... 10

Training Amazon Lex ... 12

- Documentation ... 13
- Configuration ... 13
- Testing ... 18

Training IBM Watson ... 20

- Documentation ... 20
- Configuration ... 21
- Testing ... 23

Training Rasa ... 26

- Documentation ... 26
- Configuration ... 27
- Testing ... 30

Which Chatbot Platform is Easiest to Configure? ... 32

- Winner: Documentation ... 32
- Winner: Configuration ... 32
- Winner: Testing ... 33

Winner Overall ... 35

What We Learned About Configuring a Chatbot ... 38

Which Chatbot Platforms are Easiest to Build and Train On?

Depending on your experience with machine learning, programming, and chatbots, your answer will vary. I'm here to answer this question as a Digital Marketing Specialist at Globalme by experimenting with the most popular chatbot platforms on the market. Although we do have experts that can configure these chatbots on a whim, this article will paint a picture of how easy it is to configure a chatbot with minimal experience.

For the most part, all my experience is minimal in terms of programming, machine learning, and chatbots. Although chatbots in marketing is a huge trend, marketers may not necessarily configure one from scratch. Programming? I once developed platformer game through Python for a high school project. Other than that, no experience programming at all.

Equipped with some knowledge of machine learning and natural language processing (NLP), this will be a challenging way to find an answer to our question:

Can someone that has little to no hands-on experience with conversational agents, programming, or machine learning build a chatbot?

Truth be told, we aren't building a chatbot or machine learning algorithm from scratch. Instead, we are using the framework and algorithm that each chatbot service provider uses. Even though we aren't building a chatbot, we still need to be comfortable with learning new technology.

The main challenges we are expected to face are that we might need to code and the definite need to understand the concepts behind how a chatbot learns and communicates. These are the steps we took before configuring each of the chatbots:

1. Educate ourselves and try to understand chatbots
2. Sign up for each chatbot service provider
3. If required, pick the most relevant predefined chatbot configuration
4. Make sure our goal and use case is consistent for each chatbot we configure

With a little bit of research, we embark on this journey by taking our first step: defining why we want to build a chatbot.

DEFINING THE CHATBOT’S GOAL AND USE CASE

Defining the Chatbot's Goal and Use Case

Why do we want to train a chatbot? We want to improve our customer service and increase the rate of returning customers. Traditional customer service touchpoints like replying to emails and messages manually take time. A chatbot can help customers and provide answers to their questions. In this article, we're going to create a chatbot for our widely successful business: *Quick n' Easy Projector Rentals*.

As you might have guessed, Quick n' Easy Projector Rentals is a business that rents out projectors. Our customers can be anyone from regular people who want to rent projectors for a wedding to businesses looking to hold a presentation for potential clients. To accelerate the growth of the company, our chatbot will be used to take customer inquiries and reserve projectors when needed. The end goal is to get the chatbot to say something along the lines of "Thank you. Your projector has been reserved for pick up on [insert date]." This phrase will indicate that the customer's rental was booked and that we are ready to fulfill their order.

With a chatbot, we should be able to improve our customer service and reduce the load on our human agents who currently answer the phone and email queries. The chatbot will be successfully trained if it can take an order for a **Business Projector**. To reiterate, our goal is to book rentals through this chatbot. The chatbot will be taught how to book rentals in a customer friendly way. Before we jump into building this chatbot, we need to determine what platforms we will be using.

Chatbots (or conversational agents) can be installed on social networks, messaging apps, and websites. To keep our scenario simple, our bot will be built for website use in mind. We can consider other applications like messenger apps and social media networks later. This allows us to focus on one application while also minimizing effort spent on coding. Most of the chatbot platforms in this article provide a testing environment for us, so installing programs or third-party software isn't necessary.

Mapping out a rough conversation tree

To configure our chatbot and guide it towards reaching our goal successfully, we need to plan out a typical dialogue. This is a logical first step, as it solves the IF...THEN statements that our chatbots would encounter. The key is to understand the importance of IF...THEN statements and its role in chatbot technology.

Technically it is no different than creating a dialog flow for your human customer service agents to help them navigate the calls they receive from your customers. In the end, a chatbot is a machine that will need your input and knowledge on how humans have conversations. IF a user says *this*, THEN what does your chatbot say in response?

This is a step that can make everything a tad bit easier when it comes to building our bot. Mapping out a conversation tree or dialog flow will allow you to do several things to make sure your bot is successful. Mapping out a dialog flow can help you:

- Revisit your chatbot goals and use cases
- Identify intents, entities, and the overall flow of conversations
- Provide you with a roadmap to your end goal

You don't have to spend too much time to plan this out. It can be as simple as bullet points or a mind map. For our example, our chatbot will be speaking with users who are on the *Quick n' Easy Projector Rentals* website. These users are already interested in renting a projector. Below is how a typical conversation might go between our chatbot and website visitors.

Chatbot: Hi, how can I help you?
User: I want to rent a projector
Chatbot: I can help you with that. What will you be using the projector for?
User: Presentation
Chatbot: If you need the projector for a presentation, we recommend the General Use Projector. Would you like to reserve one?
User: Yes, please
Chatbot: When would you like to pick up your projector?
User: Today
Chatbot: When would you like to return your projector?
User: Tomorrow
Chatbot: Perfect. I can reserve a Business Projector for you. It can be picked up today. Your return date will be the next day. Do you want me to reserve this projector for you?
User: Yes
Chatbot: What is your first name and email?
User: Dave and my email is dave@buyahomewithdave.com
Chatbot: Thank you. Your Business Projector has been reserved for pick up any time after 8 AM tomorrow. All booking information will be emailed to you in the next couple of minutes. If you have any questions about your rental, you can call 1-800-111-1111. See you soon!

What can we learn from the conversation? We can use the conversation tree to label variables that our chatbot needs to know. From the example conversation above, we can pull out the user intents and entities associated with them. Intents are the words that represent an action that the user wants to take. When a user says:

> **User:** I want to rent a projector

We can determine that the intent is to rent a projector. For chatbots, we would label this intent as *RentProjector*. Depending on the chatbot platform, the way we label user intents may differ. We can then link the actions that we want our chatbot to take regarding this intent. In this case, we want the chatbot to find out more information about their situation so that it can suggest the best projector for their needs.

Going through this dialog gets us to revisit our goal and map out the path towards achieving it. The conversation will be directed by the chatbot so that users will take the final step of booking a rental. This concept will be clearer when we start building our chatbots.

Did we need to create a dataset to train our chatbots? Not for this case. Our chatbot will be using small examples that we provide it with. These phrases and examples can then teach it how to respond accordingly. Depending on the platform we test, the chatbot will continue to learn and adapt to what user utterances it encounters.

Finding out which chatbot is easiest to train

This article will cover the most popular chatbot platforms on the market. The end goal is the same across all of them. We want the winning chatbot to reserve a projector rental for our users, but also do it with as little investment as possible.

After reading a few articles about configuring chatbots, we inevitably built up our chatbot vocabulary. One of the constraints we have lined up is that we will only use **a maximum of 10 custom user intents and 10 entity** to train the chatbot to do what we want it to do.

A **user intent** is the intent behind a user's typed out command. A user may say "I want to rent a projector", which tells the chatbot their intent is to...**rent a projector**.

An **entity** (or slot in some cases) provides more context to the user intent. For example, a user might type out “I want to rent a **business** projector **tomorrow**.” The entities, in this case, are the **date** that the user wants to start their rental, and the **type of projector** they want to rent.

Keep in mind we are conducting this comparison with minimal coding knowledge. As a typical marketing professional may have almost no hands-on experience with machine learning, we are in for a wild ride. To reduce the amount of time we dedicate towards training these chatbots is limited to a maximum of 3 hours per chatbot. The time will be measured from the start of training to the final delivery of the coveted sentence:

“Thank you. Your Business Projector has been reserved…”

On to our experiment.

TRAINING GOOGLE DIALOGFLOW

Training Google Dialogflow

This is the first chatbot platform we tackled in our little experiment. Google Dialogflow is an easy to understand conversational agent. Theoretically, we can have a bot up and running by understanding 3 core concepts: intents, entities, and dialog control. As stated earlier, these general concepts are followed across a majority of the chatbot platforms we played with.

Documentation

Users who are new to the chatbot space will find Dialogflow provides a great introduction to machine learning, artificial intelligence, and conversational bots. The documentation does a great job of explaining the concepts of intents, entities, and dialog control through explainer videos. There is hardly any jargon written in the “Start here” section of the documentation, and any jargon is accompanied by an explanation when needed. The Dialogflow Tutorial covers the entire chatbot process we would take to build a Dialogflow chatbot.

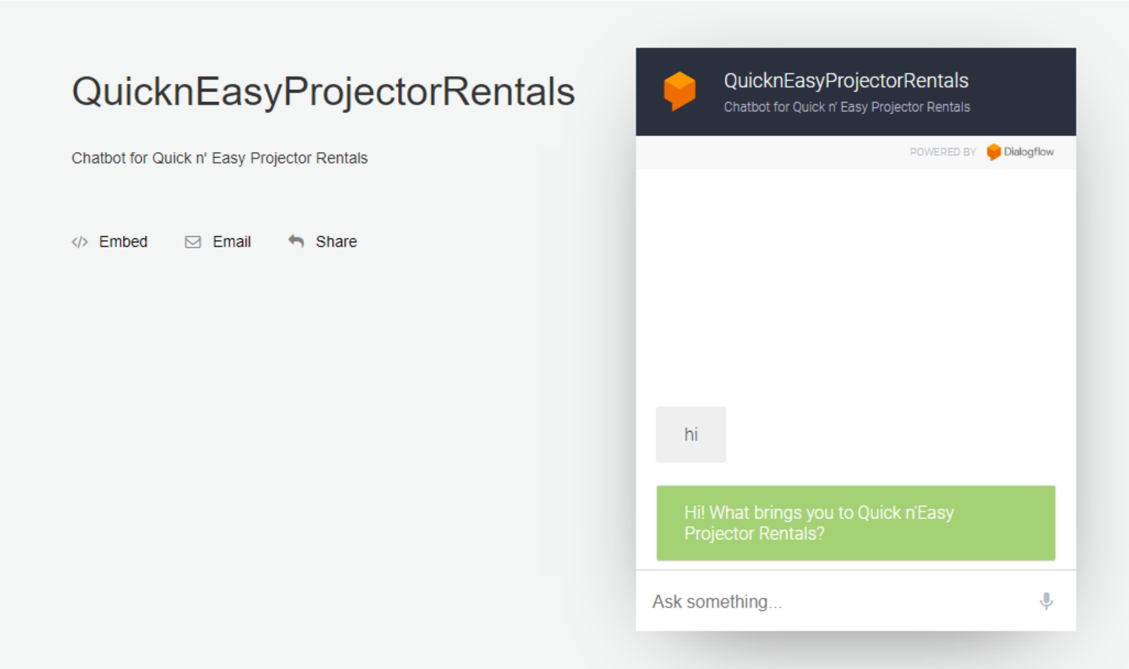
After watching the videos, glancing through the documentation, and flipping through the tutorial, we had enough information to start configuring our Dialogflow chatbot.

Configuration

Following the steps in the documentation and instructional videos, we put together a handful of user intents and entities for the agent to extract data from. The Dialogflow agent we created was also built from scratch. The intents we created ourselves are bolded below, while the unbolded intents came default with the agent.

- Default Fallback Intent
- Default Welcome Intent
- **Rental**
- **Rental - Business Projector**
- **BusinessProjector**

With a few tweaks to the Default Welcome Intent, we were able to create custom greetings to guide the way our conversation flows towards our end goal.



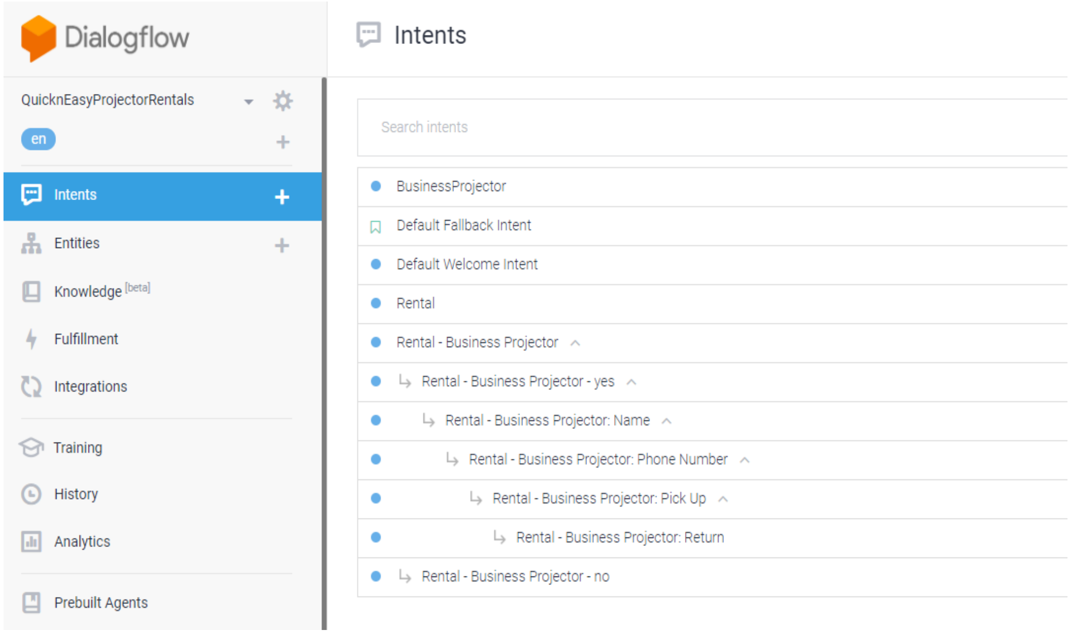
QuicknEasyProjectorRentals

Chatbot for Quick n' Easy Projector Rentals

hi

Hi! What brings you to Quick n'Easy Projector Rentals?

We wanted to add more depth to the conversation, so we had to build on the intents we mapped out. Underneath the *Rental - Business Projector* intent were several other intents to follow up on our user requests. The agent would confirm the order, collect contact details, and arrange a pickup and return date to complete our transaction. Here's what our intents look like after being fleshed out a little more.



- BusinessProjector
- Default Fallback Intent
- Default Welcome Intent
- Rental
- Rental - Business Projector
  - ↳ Rental - Business Projector - yes
    - ↳ Rental - Business Projector: Name
      - ↳ Rental - Business Projector: Phone Number
        - ↳ Rental - Business Projector: Pick Up
          - ↳ Rental - Business Projector: Return
  - ↳ Rental - Business Projector - no

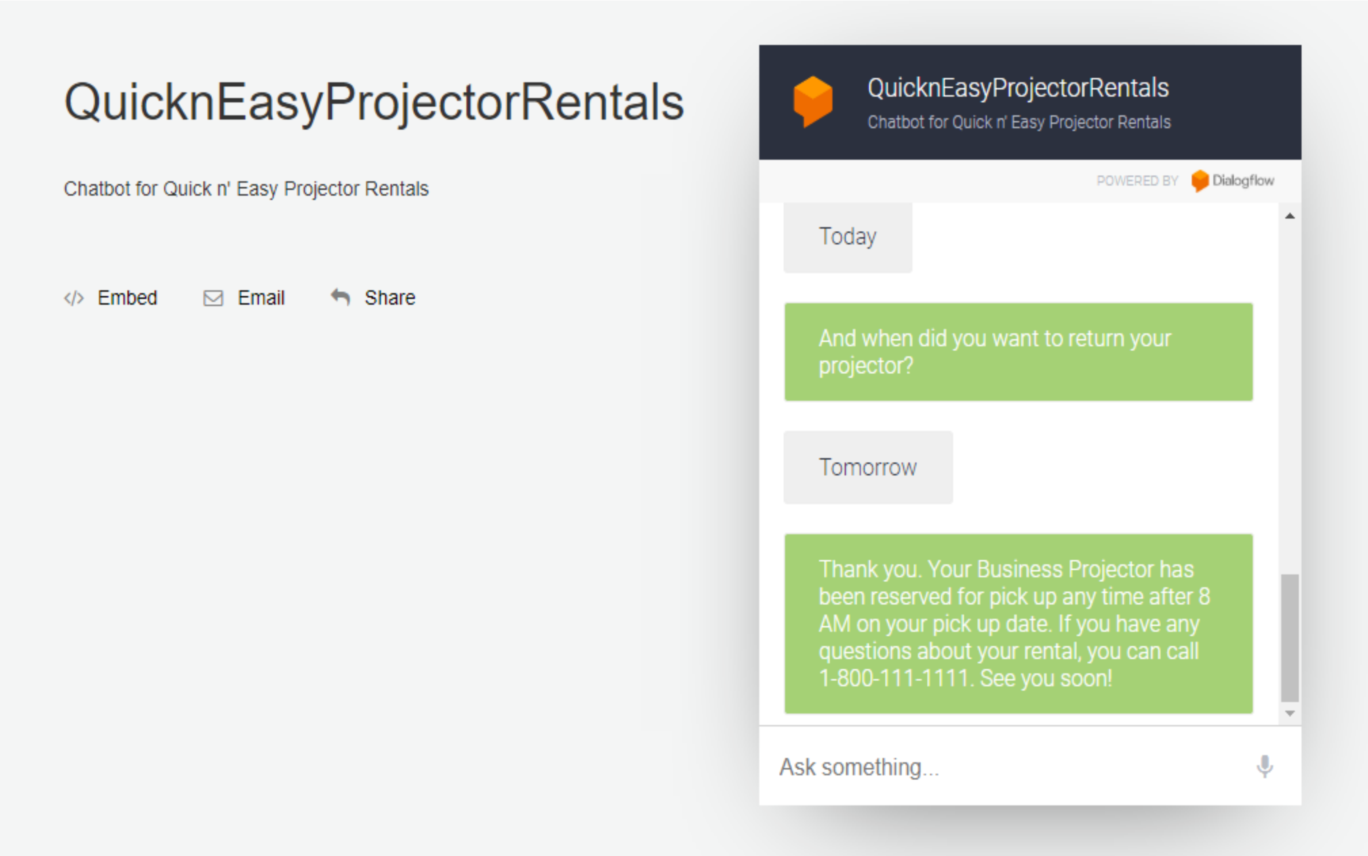
With a few configurations to the already existing framework, we made a fully functional conversational agent without breaking a sweat.

Testing

It took 3 hours to train and fully configure Google Dialogflow to do what we needed, in addition to a few more bells and whistles. It only took the Google Dialogflow chatbot agent 9 custom intents and 2 entities to understand our user commands.

We were able to successfully book our Business Projector rental with ease. Although the fulfillment and integration options haven't been configured to push these reservations to a booking system, we were able to make our reservation for a Business Projector.

The end result?


QuicknEasyProjectorRentals

Chatbot for Quick n' Easy Projector Rentals

QuicknEasyProjectorRentals

Chatbot for Quick n' Easy Projector Rentals

Today

And when did you want to return your projector?

Tomorrow

Thank you. Your Business Projector has been reserved for pick up any time after 8 AM on your pick up date. If you have any questions about your rental, you can call 1-800-111-1111. See you soon!

Ask something...

TRAINING AMAZON LEX

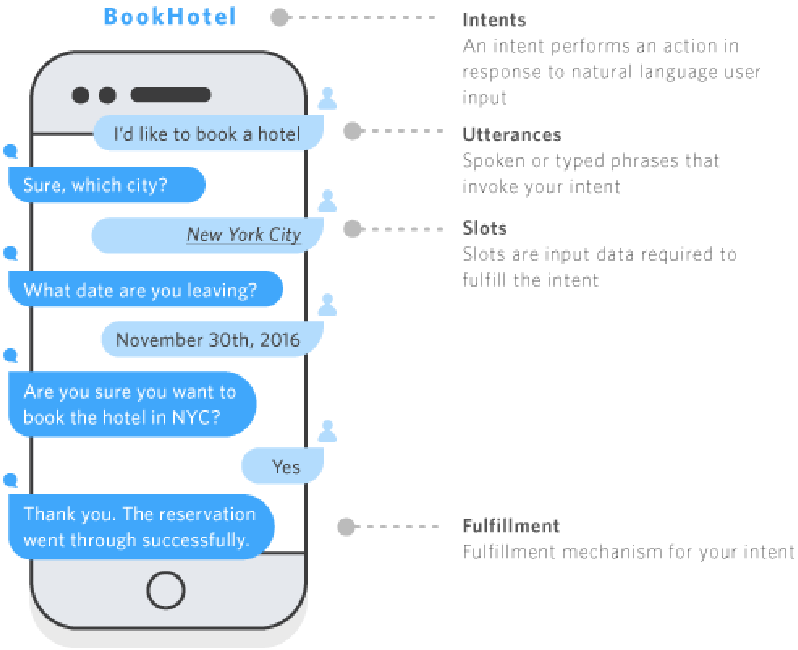
Training Amazon Lex

Amazon Lex

Amazon Lex is a service for building conversational interfaces using voice and text. With Lex, the same deep learning engine that powers Alexa is now available to any developer, enabling you to bring sophisticated, natural language chatbots to your new and existing applications.

Amazon Lex is simple and easy to start from scratch with - much like Google Dialogflow. We are given the option to get a conversational chatbot started right away. Amazon Lex gives off a more complex feel to it, which may appeal to more advanced users.

Amazon Lex uses automatic speech recognition (ASR) and natural language understanding (NLU) to enable developers to build user-friendly chatbots. We don't have to understand these two concepts in detail, but Amazon Lex does use the same variables that most of the other chatbot platforms use: **intents**, **utterances**, and **slots**. Amazon provides the image below to explain how Amazon Lex should work.



I'd like to book a hotel

Sure, which city?

New York City

What date are you leaving?

November 30th, 2016

Are you sure you want to book the hotel in NYC?

Yes

Thank you. The reservation went through successfully.

Intents
An intent performs an action in response to natural language user input

Utterances
Spoken or typed phrases that invoke your intent

Slots
Slots are input data required to fulfill the intent

Fulfillment
Fulfillment mechanism for your intent

Documentation

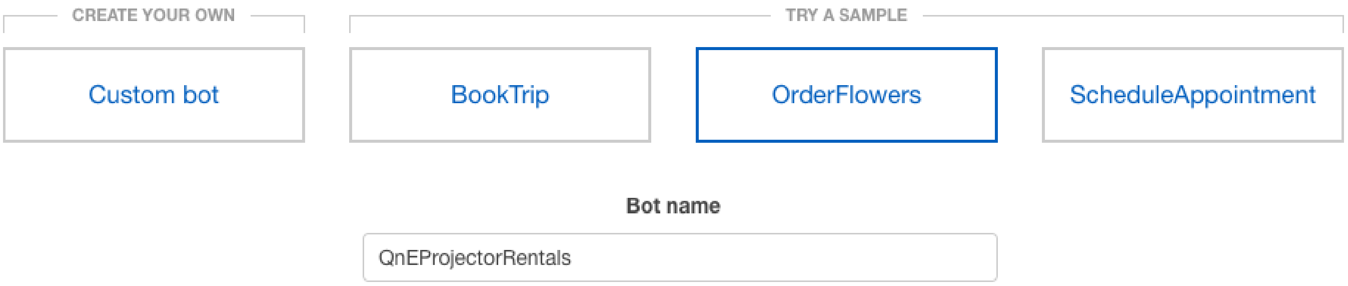
The written documentation can get fairly complex, which is why we had to watch the Amazon Lex introduction videos instead. Watching the videos gave a more user-friendly explanation of how a chatbot can be built with Amazon Lex. Since we started with Google Dialogflow, it was somewhat easier to understand the Amazon Lex documentation and concepts described.

What makes their documentation different from the other platforms is that it is oriented towards more advanced users. For users who are more familiar with machine learning technologies and chatbots, the Amazon Lex documentation gets into advanced concepts like Lambda Functions and the platform's deep learning technology.

After setting up our AWS account, we were able to begin configuring our Amazon Lex bot quickly.

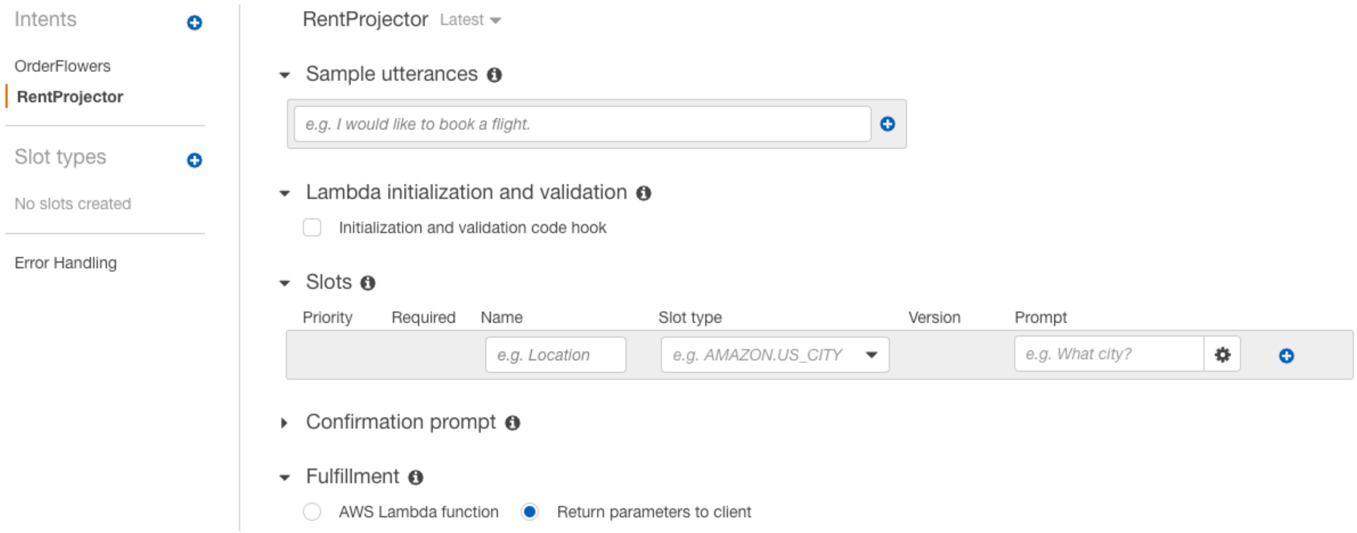
Configuration

What made the entire process easier was the option to choose a bot that Amazon has prebuilt. These examples can give us a foundation towards creating a more built out and functional chatbot. In our case we want our Quick n' Easy Projector Rentals customers to book a rental through the chatbot. We chose the OrderFlowers Amazon Lex sample since it sounded like it would be the closest to our end product.



Bot name: QnEProjectorRentals

Right off the bat, we are greeted with a simple workspace that has an overview of the different components that would power our own Amazon Lex bot. Sample utterances, slots, user request validation, fulfillment, and responses can be adjusted on a whim. Utterances and responses are simple to understand. Utterances are what the user says, whereas responses are phrases the chatbot responds with. We can build a list of intents to customize the chatbot to handle projector related queries.

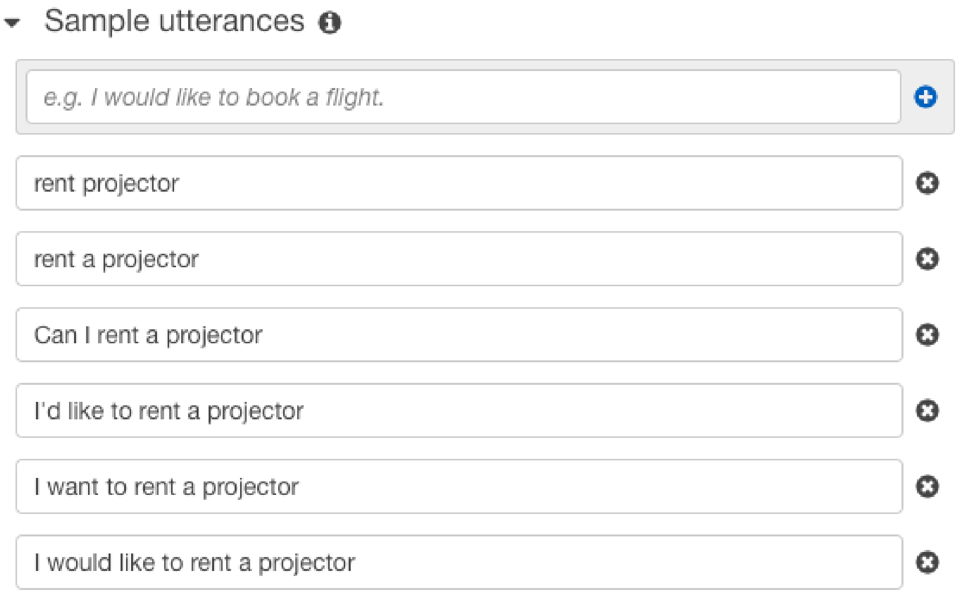


Slots are what we would call "entities" for other conversational bot platforms. These are what Amazon Lex will use to decipher the sample utterances. This is where the data aspect of chatbots comes into play. Having more slots means the bot will be extracting or recognizing data points like the user's desired ProjectorType, PickupDate, or PickupTime. The slots are also what the user needs to specify in a sentence so that the chatbot can identify the intent.

We've created two slot types and intents to make sure our conversational agent can understand how to respond in a scenario where a user wants to rent a projector for their business presentation. We only needed two intents for this scenario. The two intents we created were:

- RentProjector
- RentBusinessProjector

RentProjector would tell Amazon Lex whether a user was interested in renting a projector. This is the intent that would initiate the entire rental conversation. This intent had 6 sample utterances and 1 slot type. The sample utterances were variations of "can I rent a projector" or "I want to rent a projector."



Sample utterances

e.g. I would like to book a flight.

rent projector

rent a projector

Can I rent a projector

I'd like to rent a projector

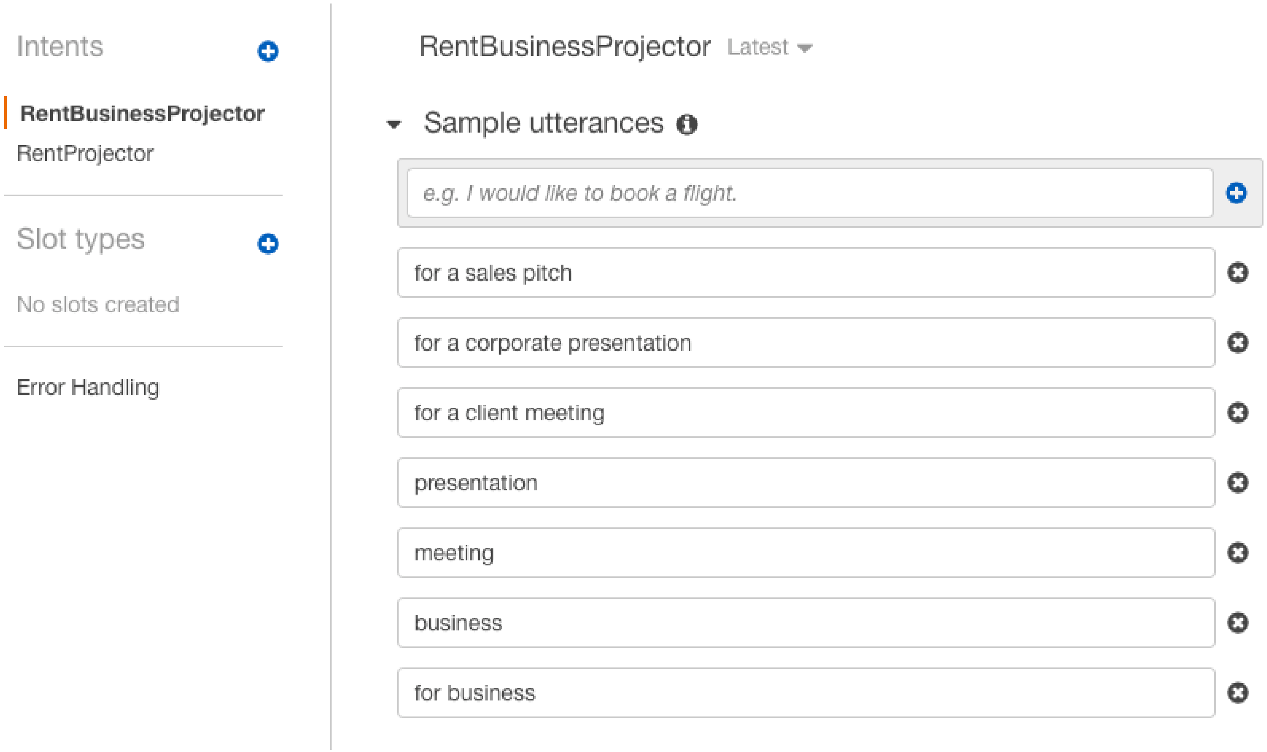
I want to rent a projector

I would like to rent a projector

The RentProjector intent was linked to a custom slot we created which we named BusinessProjector. Since we want the user to feel like they have a more tailored experience, we gave the slot the prompt of "What will you use the projector for?" This isn't normally how you would configure a slot like this, but in our case, this is what worked. We are testing to see if we can get the conversational bot to take an order for a Business Projector.

This slot would automatically expand on values like "presentation", "business", or "meeting" to direct the conversation towards the next intent - booking a Business Projector. Although we listed only a handful of values in this slot, we could definitely expand it in the future with a more thought out dataset.

Our second intent was named RentBusinessProjector and was mapped out to respond to business projector related utterances. Notice how there aren't any custom slot types for this intent.


Intents

RentBusinessProjector

RentProjector

Slot types

No slots created

Error Handling

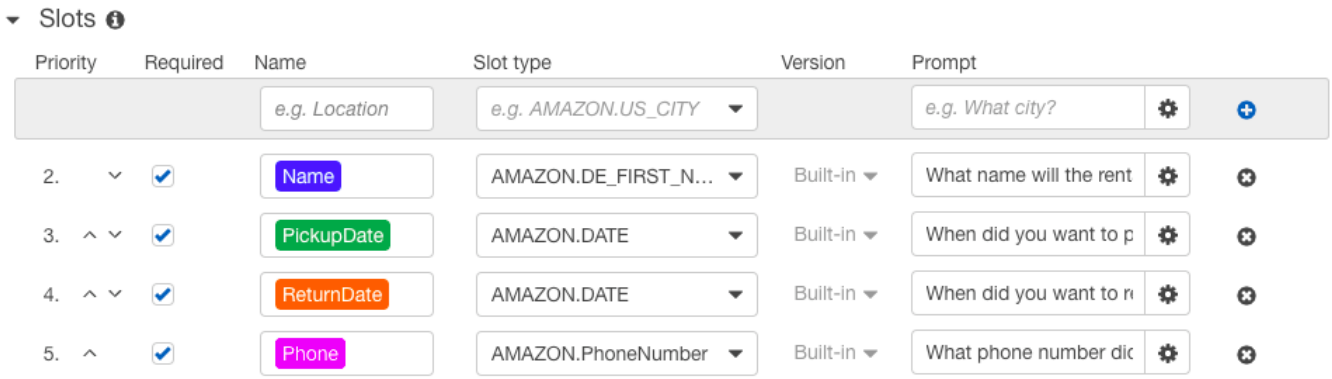
RentBusinessProjector Latest

Sample utterances

e.g. I would like to book a flight.

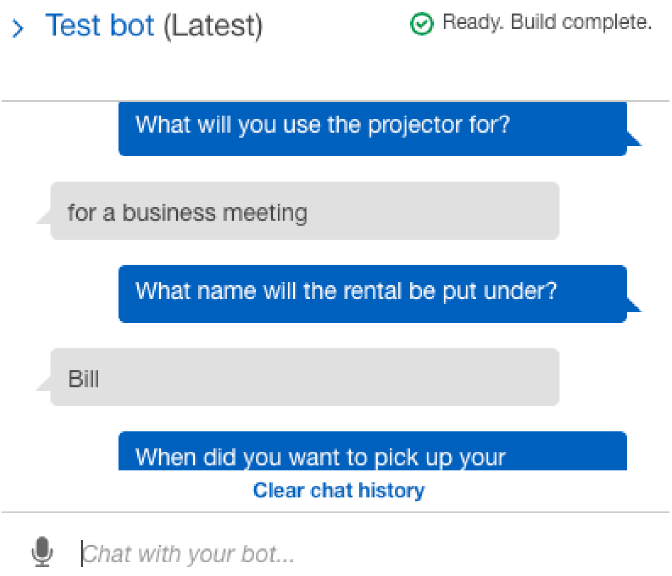
- for a sales pitch
- for a corporate presentation
- for a client meeting
- presentation
- meeting
- business
- for business

Amazon has already provided a list of slot types we could use so that the bot can extract the data we require to book a rental. Variables such as the name, pick up and return dates, and the customer's phone number was easy to set up through the prebuilt slots. The bot would prompt the user at each step in the conversation and collect these data points as needed. Since this is a transactional conversation and requires all the client's information, we hit require on all the slot types.


Slots

Priority	Required	Name	Slot type	Version	Prompt
		e.g. Location	*e.g. AMAZON.US_CITY*		*e.g. What city?*
2.	☑	Name	AMAZON.DE_FIRST_N...	Built-in	What name will the rent
3.	☑	PickupDate	AMAZON.DATE	Built-in	When did you want to p
4.	☑	ReturnDate	AMAZON.DATE	Built-in	When did you want to r
5.	☑	Phone	AMAZON.PhoneNumber	Built-in	What phone number dic

These slots were simple to configure. Here's what happens when we carry on a conversation with the intent to rent a business projector.



What will you use the projector for?

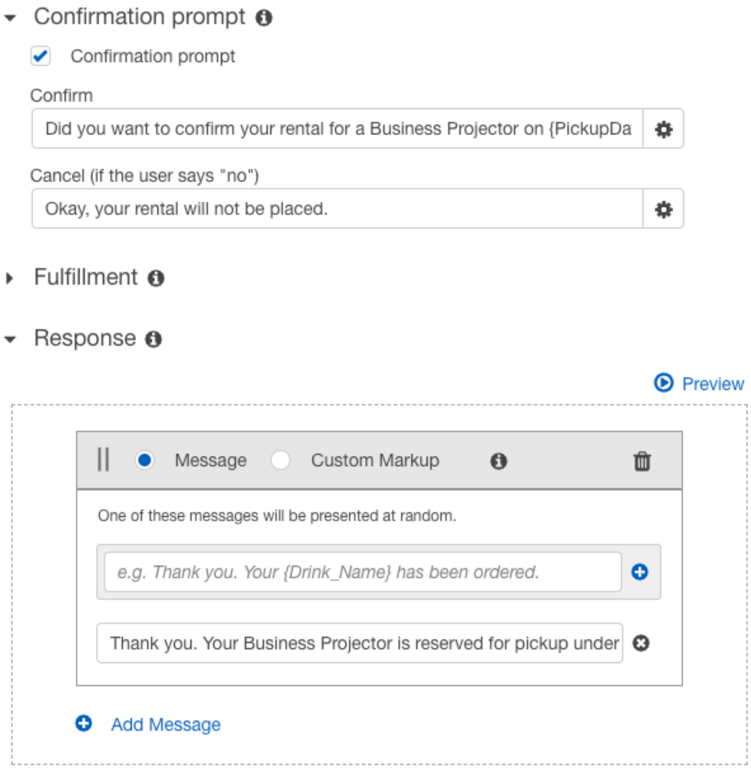
for a business meeting

What name will the rental be put under?

Bill

When did you want to pick up your

To further increase our user's experience, we enabled a couple more options. A confirmation prompt and ending response were added to close the RentBusinessProjector intent in a user-friendly way.



☑ Confirmation prompt

Confirm

Did you want to confirm your rental for a Business Projector on {PickupDa

Cancel (if the user says "no")

Okay, your rental will not be placed.

● Message ○ Custom Markup

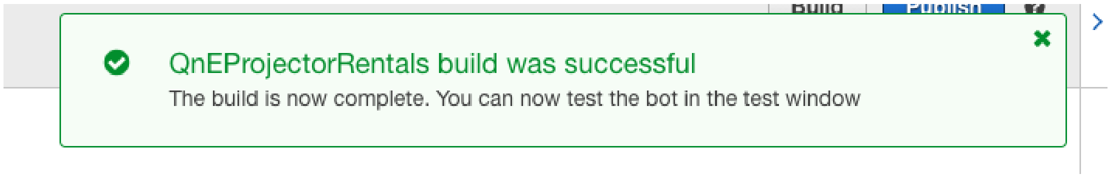
One of these messages will be presented at random.

e.g. Thank you. Your {Drink_Name} has been ordered.

Thank you. Your Business Projector is reserved for pickup under

Testing

How did our configuration go?

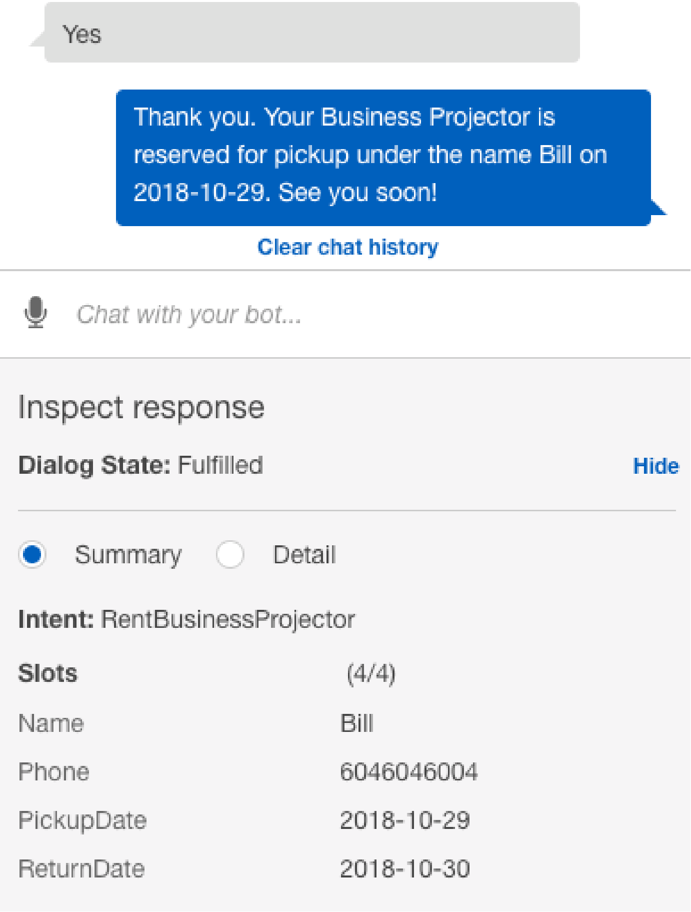


QnEProjectorRentals build was successful

The build is now complete. You can now test the bot in the test window

It took only 2 hours to configure our Amazon Lex conversational bot. The bot was able to take our order and grab all information to make sure we can fulfill this rental on our end. In terms of getting this bot to act as a front-end conversational agent, we can manage the integrations under the *Channel* settings in the dashboard. The chatbot can be enabled on Facebook, Kik, Slack, and Twilio.

Integrating with these chat platforms can be done quite easily, however, this is where the limitation lies. We won't be able to use Amazon Lex on our website without customization.



Yes

Thank you. Your Business Projector is reserved for pickup under the name Bill on 2018-10-29. See you soon!

Inspect response

Dialog State: Fulfilled

● Summary ○ Detail

Intent: RentBusinessProjector

Slots	(4/4)
Name	Bill
Phone	6046046004
PickupDate	2018-10-29
ReturnDate	2018-10-30

TRAINING IBM WATSON

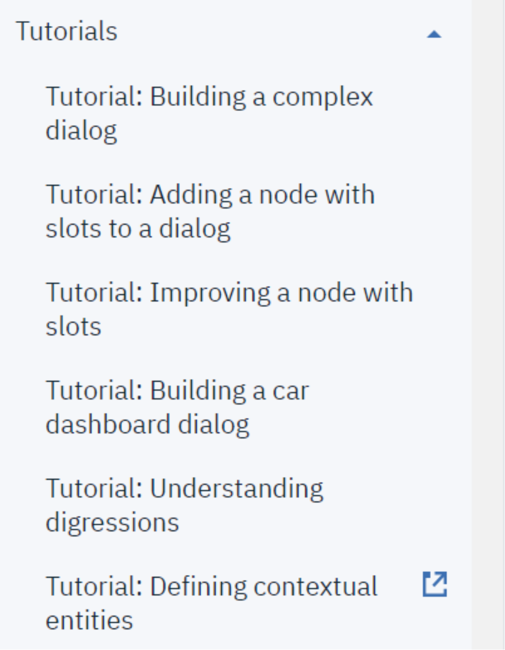
Training IBM Watson

IBM Watson is touted as a question-and-answer system that can be used to build applications and chatbots. The IBM Watson platform allows us to create an application that shares a dialog interaction between our chatbot and users on Quick n' Easy Projector Rentals. The IBM interface is simple to use, and no back-end coding is shown at first glance. The chatbot can be easily integrated into other applications such as Slack, Facebook, and Twilio.

The IBM Watson is feature packed and powerful enough to be an enterprise-level solution, but simple and easy to understand for those who are looking for an entry-level platform to build a chatbot.

Documentation

For this article, we followed the Getting Started Tutorial. The documentation covered each step thoroughly with the help of labeled graphics. Following this tutorial gave us the ability to create a Watson Assistant with ease. There are also tutorials for more intricate Watson assistant functions, such as building a complex dialog.

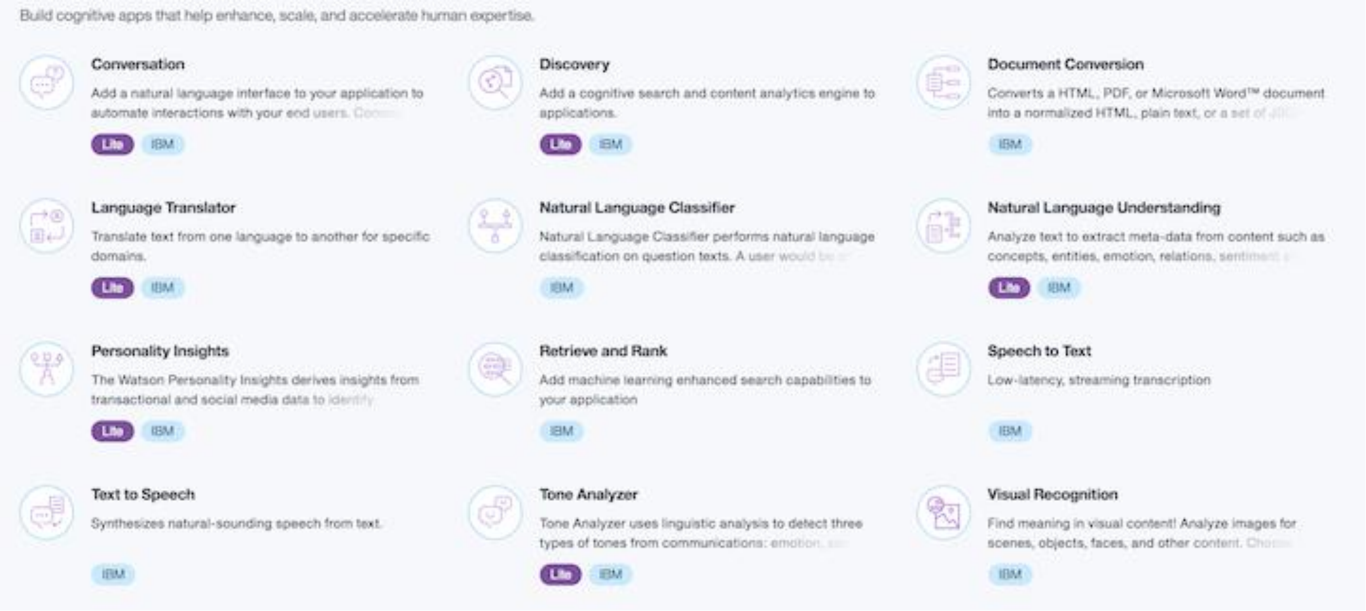


Tutorials
Tutorial: Building a complex dialog
Tutorial: Adding a node with slots to a dialog
Tutorial: Improving a node with slots
Tutorial: Building a car dashboard dialog
Tutorial: Understanding digressions
Tutorial: Defining contextual entities

IBM provides thorough documentation without overwhelming first-time users. We have already looked over both Amazon Lex and Google Dialogflow documentations at this point. This enabled us to have a better understanding of the IBM Watson documentation, as well as a better foundation for configuring IBM Watson.

Configuration

We started off this small trial by creating an IBM Watson account and picking between the many examples of chat assistants prebuilt by IBM. With all of the options laid out in front of us, the best option for our scenario is the “conversation” application.



Build cognitive apps that help enhance, scale, and accelerate human expertise.

Conversation

Add a natural language interface to your application to automate interactions with your end users.

Discovery

Add a cognitive search and content analytics engine to applications.

Document Conversion

Converts a HTML, PDF, or Microsoft Word™ document into a normalized HTML, plain text, or a set of

Language Translator

Translate text from one language to another for specific domains.

Natural Language Classifier

Natural Language Classifier performs natural language classification on question texts. A user would be

Natural Language Understanding

Analyze text to extract meta-data from content such as concepts, entities, emotion, relations, sentiment

Personality Insights

The Watson Personality Insights derives insights from transactional and social media data to identify

Retrieve and Rank

Add machine learning enhanced search capabilities to your application

Speech to Text

Low-latency, streaming transcription

Text to Speech

Synthesizes natural-sounding speech from text.

Tone Analyzer

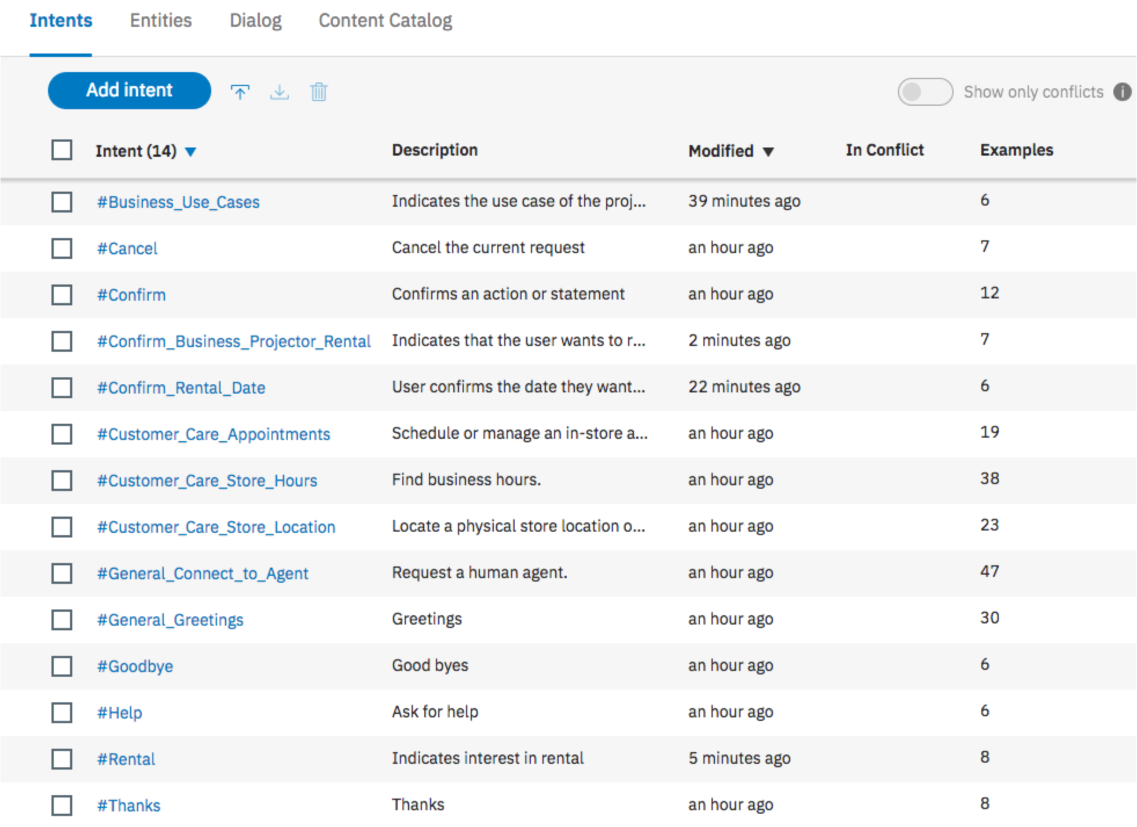
Tone Analyzer uses linguistic analysis to detect three types of tones from communications: emotion,

Visual Recognition

Find meaning in visual content! Analyze images for scenes, objects, faces, and other content.

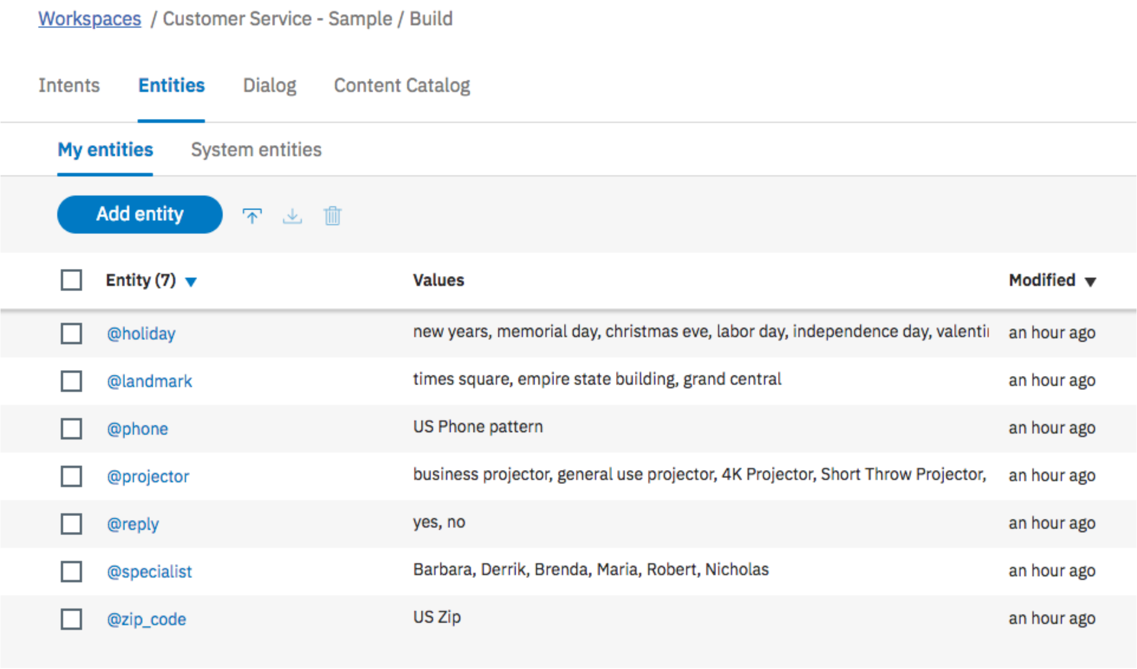
This application was prebuilt to include predefined intents, entities, and dialog. These intents were packaged with up to 36 different user examples so that IBM Watson would work right out of the box as a conversation chatbot. Intents as simple as *#General_Greetings* were included with more complicated intents like *#Customer_Care_Appointments*. The only other intents we had to add to get this chatbot to work for us were:

- #Business_Use_Cases: Words and phrases that indicate the user would find renting the Business Projector most useful
- #Rental: Key phrases that indicate the user wants to initiate a conversation about rentals
- #Confirm: Words and phrases used to confirm a statement
- #Confirm_Business_Projector_Rental:
  Words and phrases that indicate that the user wants to rent a business projector
- #Confirm_Rental_Date: Words and phrases used to establish dates regarding rentals



☐	Intent (14) ▼	Description	Modified ▼	In Conflict	Examples
☐	#Business_Use_Cases	Indicates the use case of the proj...	39 minutes ago		6
☐	#Cancel	Cancel the current request	an hour ago		7
☐	#Confirm	Confirms an action or statement	an hour ago		12
☐	#Confirm_Business_Projector_Rental	Indicates that the user wants to r...	2 minutes ago		7
☐	#Confirm_Rental_Date	User confirms the date they want...	22 minutes ago		6
☐	#Customer_Care_Appointments	Schedule or manage an in-store a...	an hour ago		19
☐	#Customer_Care_Store_Hours	Find business hours.	an hour ago		38
☐	#Customer_Care_Store_Location	Locate a physical store location o...	an hour ago		23
☐	#General_Connect_to_Agent	Request a human agent.	an hour ago		47
☐	#General_Greetings	Greetings	an hour ago		30
☐	#Goodbye	Good byes	an hour ago		6
☐	#Help	Ask for help	an hour ago		6
☐	#Rental	Indicates interest in rental	5 minutes ago		8
☐	#Thanks	Thanks	an hour ago		8

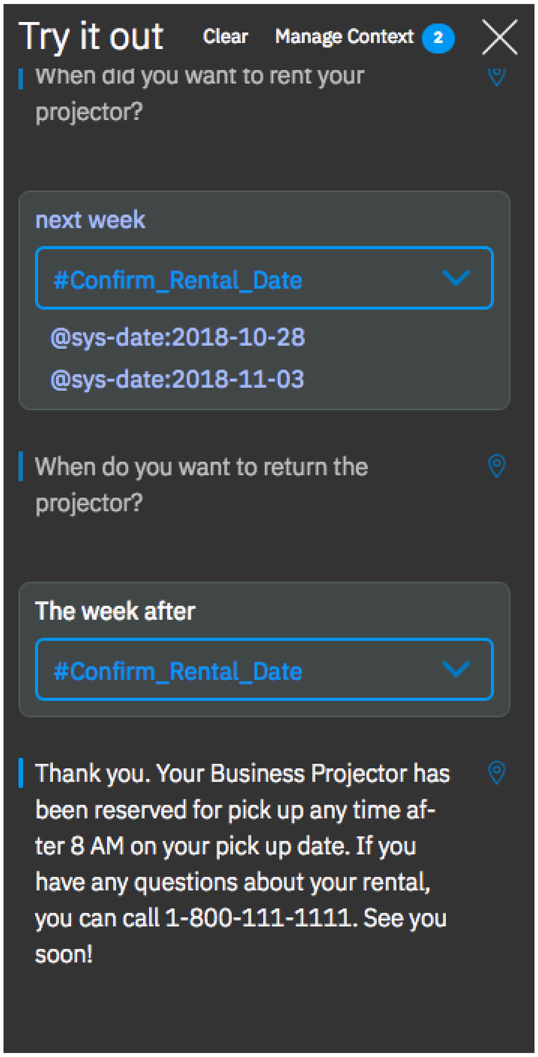
The entities that were prebuilt and preloaded made the IBM Watson conversational agent easier to develop. We only needed to make 1 custom entity: *@projector*. All other entities were added as defaults.



☐	Entity (7) ▼	Values	Modified ▼
☐	@holiday	new years, memorial day, christmas eve, labor day, independence day, valenti	an hour ago
☐	@landmark	times square, empire state building, grand central	an hour ago
☐	@phone	US Phone pattern	an hour ago
☐	@projector	business projector, general use projector, 4K Projector, Short Throw Projector,	an hour ago
☐	@reply	yes, no	an hour ago
☐	@specialist	Barbara, Derrik, Brenda, Maria, Robert, Nicholas	an hour ago
☐	@zip_code	US Zip	an hour ago

Testing

The result after only 5 custom intents and 1 custom entity? It took 1.5 hours to build and train IBM Watson towards achieving our goal. This gave us loads of spare time to include other entities and intents to continue developing our IBM Watson further. Here's how a conversation went when we introduced a few more intents and projector types in the dialog flow.



When did you want to rent your projector?

next week

#Confirm_Rental_Date

@sys-date:2018-10-28

@sys-date:2018-11-03

When do you want to return the projector?

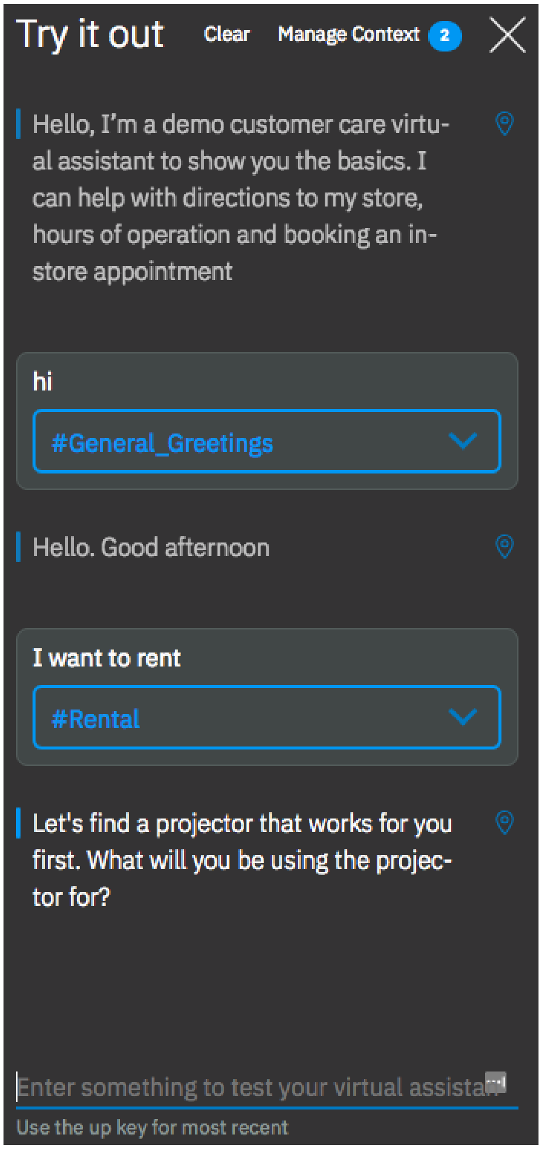
The week after

#Confirm_Rental_Date

Thank you. Your Business Projector has been reserved for pick up any time after 8 AM on your pick up date. If you have any questions about your rental, you can call 1-800-111-1111. See you soon!

With an additional 30 minutes of training and tinkering around, IBM Watson is able to deliver responses to typical commands from our Quick n' Easy Projector Rentals audience. A few tweaks to the already existent IBM's customer service assistant gave us a functional and user-friendly chatbot that took our projector order successfully.

Even with a small dataset of example phrases and intents, our Watson chatbot was able to understand natural conversations like the one below. Watson can learn from small datasets which require very little investment to create. As we mentioned earlier, we wanted to limit the time spent feeding example phrases to each chatbot. IBM Watson shows that when it comes to performance, nailing the quality of your data is more important than the quantity of your data.


Try it out
Clear
Manage Context 2

Hello, I'm a demo customer care virtual assistant to show you the basics. I can help with directions to my store, hours of operation and booking an in-store appointment

hi

#General_Greetings

Hello. Good afternoon

I want to rent

#Rental

Let's find a projector that works for you first. What will you be using the projector for?

Enter something to test your virtual assista
Use the up key for most recent

TRAINING RASA

Training Rasa

Rasa is an open source chatbot that is equipped with a natural language processing tool. The open source tool is called Rasa NLU. You can tweak and customize the machine learning algorithm that Rasa uses so that you can create a model that provides the results you desire. Rasa NLU can be run wherever you want it to, and none of your training data has to be passed over to Google, Microsoft, Amazon, or Facebook to train your chatbot.

Documentation

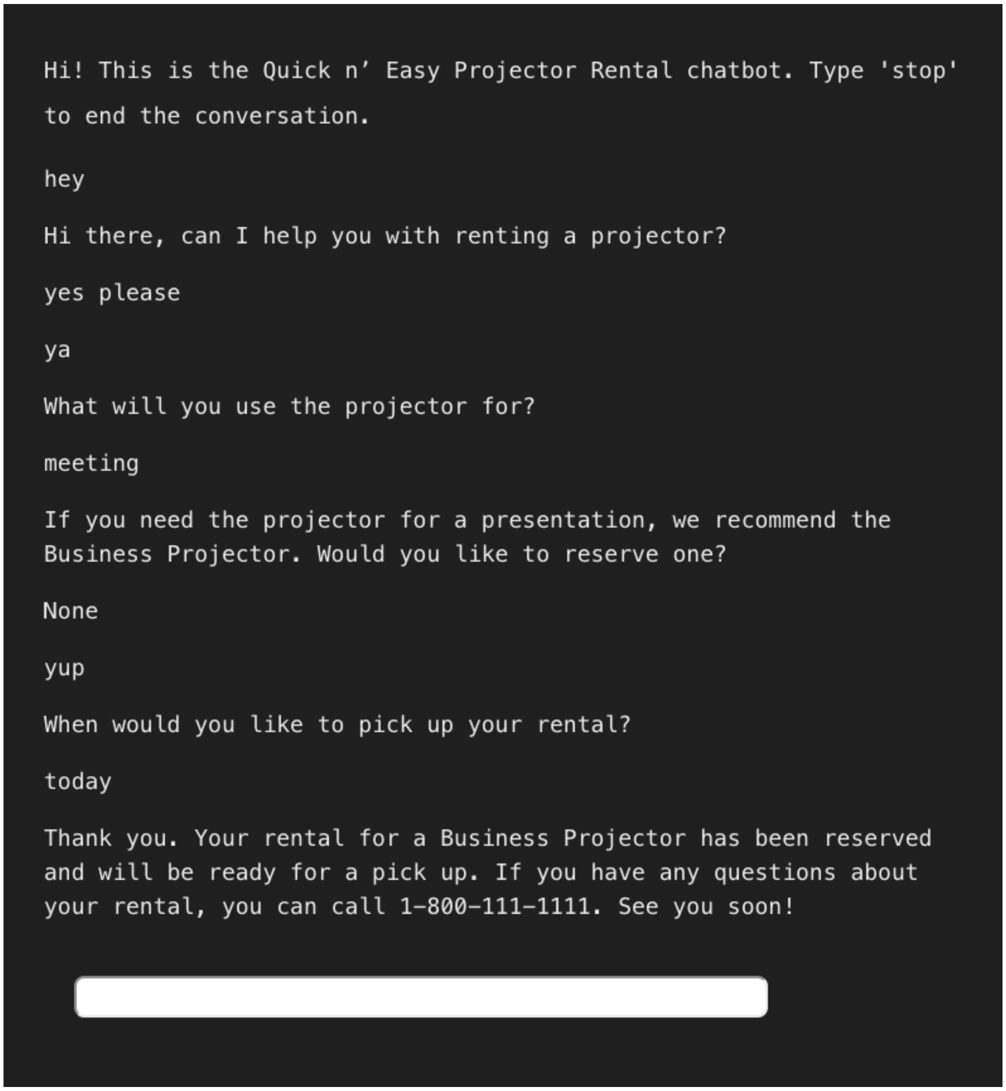
The Rasa chatbot platform is extremely easy to understand for a platform that requires hands-on coding. In just 6 steps, we can have a fully functioning chatbot. Well, this is what we have gathered from the documentation. For our purpose, we won't need to delve into the nitty-gritty details of the chatbot architecture. These are the steps that we will follow which are outlined in the Rasa quick start guide.

1. Write stories
2. Define a domain
3. Train a dialog model
4. Talking to your bot
5. Add NLU
6. Talking to your bot

With these steps in mind, we should be able to put together a bot that reserves our projector rental relatively easily. Of all the chatbot platforms we've worked with, this one requires the most engagement and coding experience.

Configuration

Here's what we were able to put together in a 3-hour time limit:



```
Hi! This is the Quick n’ Easy Projector Rental chatbot. Type 'stop'
to end the conversation.

hey

Hi there, can I help you with renting a projector?

yes please

ya

What will you use the projector for?

meeting

If you need the projector for a presentation, we recommend the
Business Projector. Would you like to reserve one?

None

yup

When would you like to pick up your rental?

today

Thank you. Your rental for a Business Projector has been reserved
and will be ready for a pick up. If you have any questions about
your rental, you can call 1-800-111-1111. See you soon!
```

The above took about 2.5 hours to train and configure for Quick n' Easy Projector Rentals. It's not the prettiest or most robust result, but it is ready to be rolled out as a Facebook messenger chatbot. So, what did we find?

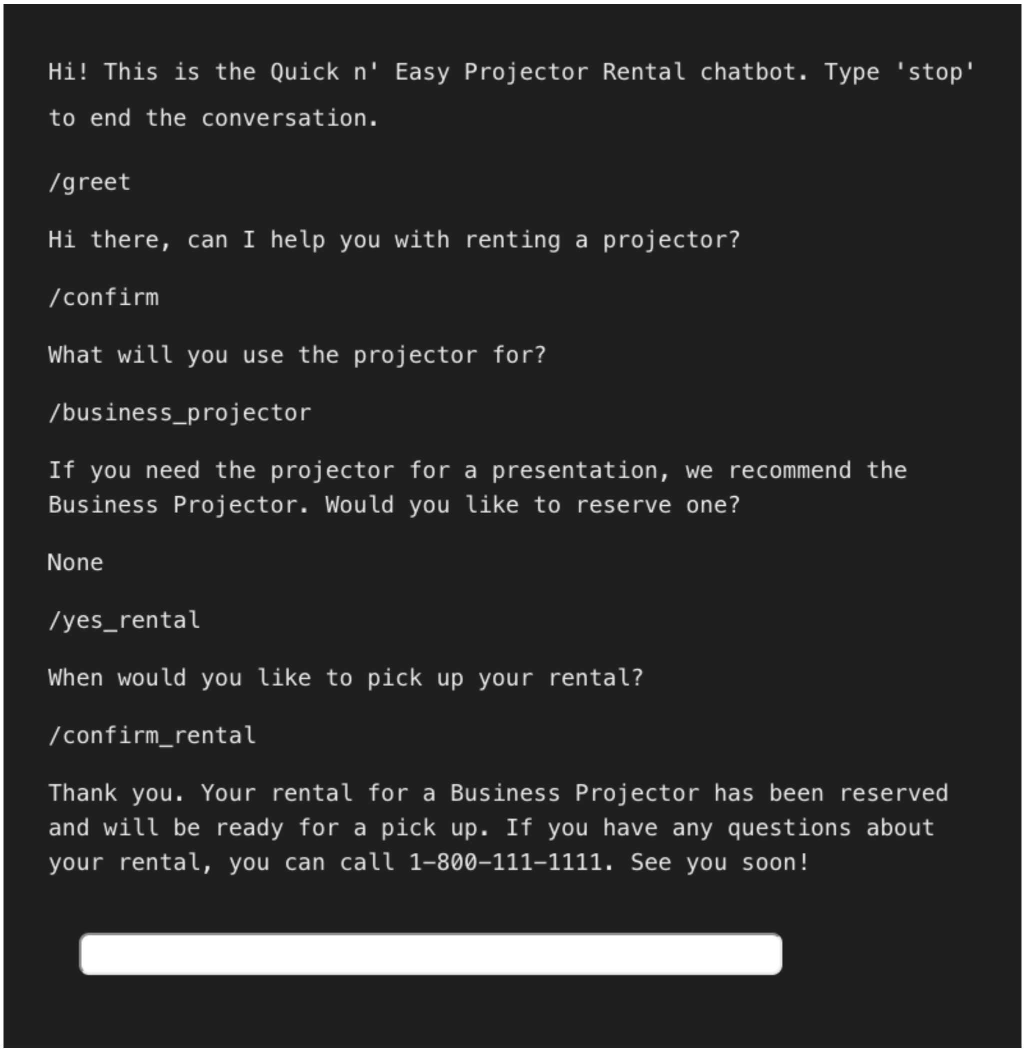
For someone with almost zero coding experience, this was simple to do. Much of the required Python coding was included in the Quick Start section, and it only took some adjustments and tweaking to get it to work for our Quick n' Easy Projectors business example. We were also given a testing environment to make things simpler. The only gripe with this was having to refresh the page every time we wanted to rerun the code for the chatbot.

After reading the documentation and playing around with the example that Rasa provided, we were able to build a simple chatbot. Following the 6 steps gave us a step by step process towards learning how Rasa can be programmed to react based on custom user intents, its own actions, and phrase templates.

The user intents we had mapped out for Rasa were:

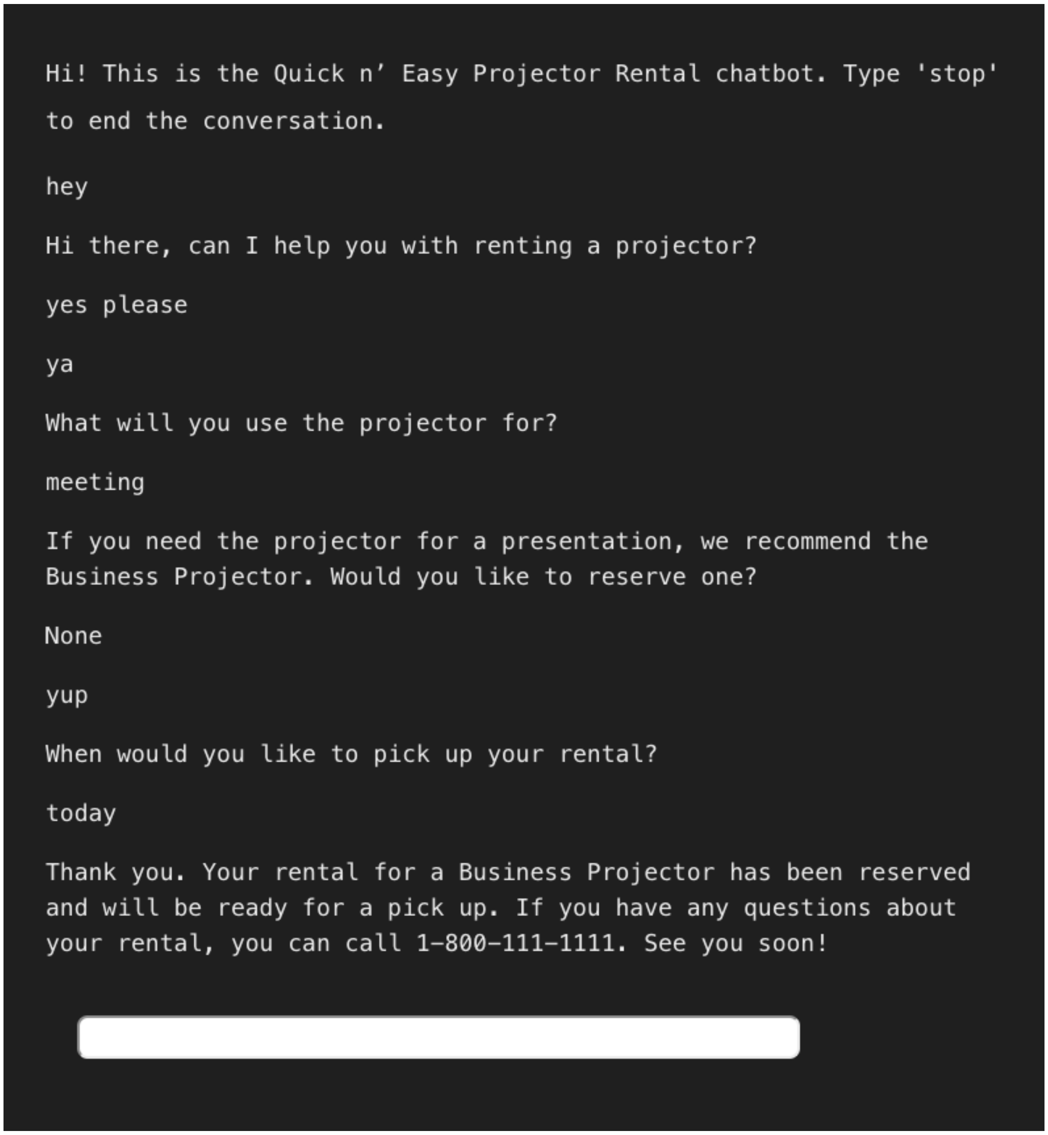
- **greet:** the user says something to acknowledge the bot
- **confirm:** the user confirms they want to rent a projector
- **business_projector:** user indicates that they want to use the projector for a business presentation
- **yes_rental:** user confirms that they want to rent the projector
- **confirm_rental:** user confirms they want to rent the projector by specifying a pickup date

The screenshot below shows how each intent gets mapped out to the way the chatbot responds. These intents haven't been associated with words yet. The Rasa NLU tool allows us to link an intent such as *confirm*, with natural sounding words like "yes" or "yup."



```
Hi! This is the Quick n' Easy Projector Rental chatbot. Type 'stop'
to end the conversation.

/greet

Hi there, can I help you with renting a projector?

/confirm

What will you use the projector for?

/business_projector

If you need the projector for a presentation, we recommend the
Business Projector. Would you like to reserve one?

None

/yes_rental

When would you like to pick up your rental?

/confirm_rental

Thank you. Your rental for a Business Projector has been reserved
and will be ready for a pick up. If you have any questions about
your rental, you can call 1-800-111-1111. See you soon!
```

When we interacted with the bot and engaged in a simple conversation, we got the desired result for the chatbot to say “Thank you. Your rental for a Business Projector has been reserved and will be ready for a pickup.”



```
Hi! This is the Quick n' Easy Projector Rental chatbot. Type 'stop'
to end the conversation.

hey

Hi there, can I help you with renting a projector?

yes please

ya

What will you use the projector for?

meeting

If you need the projector for a presentation, we recommend the
Business Projector. Would you like to reserve one?

None

yup

When would you like to pick up your rental?

today

Thank you. Your rental for a Business Projector has been reserved
and will be ready for a pick up. If you have any questions about
your rental, you can call 1-800-111-1111. See you soon!
```

Testing

From the last image, you may have noticed that the chatbot didn't understand our initial "yes please" input. Although the words "yes" and "please" were assigned to the *confirm* intent, the chatbot was not able to pick this up. With a little more time, we could have got this input to work.

You might also have noticed that the Rasa chatbot spits out the word "None" after it has responded with "...Would you like to reserve one?" For some reason, this would spit out no matter how many times we tried to alter the code.

Did the chatbot take our order successfully? **Yes**, although it was very rigid in how it could be programmed, the chatbot still took our order successfully. It took about 2.5 hours to build and required only 5 intents to be written.

WHICH PLATFORM WAS EASIEST TO CONFIGURE?

Which Chatbot Platform Was Easiest to Configure?

Truth be told, the big three chatbot platforms: Amazon, Google, and IBM were all very similar in experience. Rasa was the only chatbot platform that was tested which required some hands-on Python coding.

Winner: Documentation

Reading the documentation for each platform was simple enough to follow, and each one of them provided tutorials. We spent an average of 1 hour reading the documentation and terminologies for each platform. Rasa had the most interactive documentation out of the entire group. Rasa allowed users to play with the Python code in the Get Started guide, which followed a story-like framework.

Although Rasa had the most entertaining documentation, IBM Watson and Google Dialogflow seemed to have the more thorough yet simple resources. Comparing IBM Watson and Google Dialogflow resources were close. Both gave a great explanation of how their platforms worked, but Google Dialogflow explained difficult concepts with high-quality visual aids and videos. These videos and the simple wording in the documentation gave us the foundation to understand all the other platforms with ease.

For novice chatbot explorers - Google Dialogflow's documentation wins.

Winner: Configuration

Comparing the configuration of each platform was more difficult. Each one provided a different experience, and each one had its quirks. One thing's for sure: The Rasa chatbot was more difficult to configure on the fly. It wasn't as simple as the other platforms, but we can appreciate that it still made configuring a chatbot with Python an easy experience.

Amongst Amazon Lex, Google Dialogflow, and IBM Watson platforms, Amazon and IBM did a great job of building examples that we could adapt for our use case. It took us 2 hours to configure Amazon Lex, and just over 1.5 hours to fully configure IBM Watson.

Why did IBM Watson take only 1.5 hours? They provided enough examples of user intents, entities, and natural language data for our chatbot to be fully configured to our needs. In fact, we dedicated another half an hour to configure our IBM Watson bot a little more. It felt so much more intuitive configuring IBM Watson versus Google Dialogflow. Amazon Lex falls between the two in terms of the configuration experience.

The IBM Watson conversational agent takes the cake in our configuration experience.

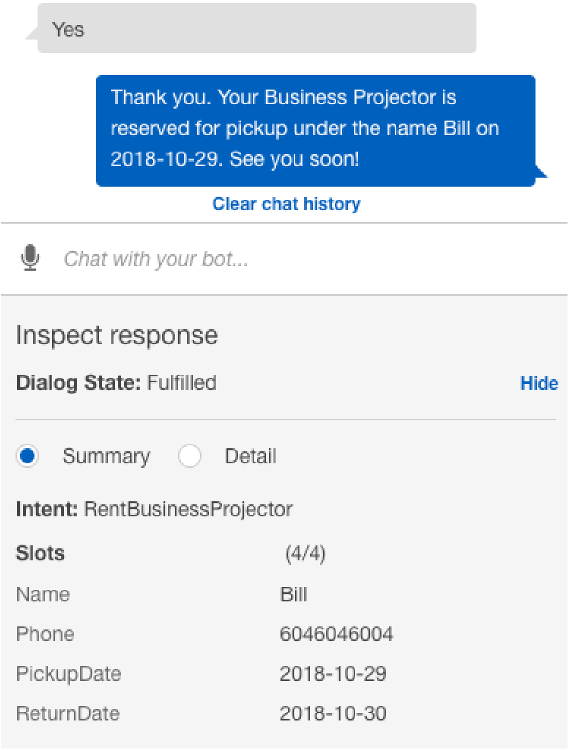
Winner: Testing

Testing the chatbots we configured allowed us to detect the elements we needed to work on, as well as any glaring mistakes. Let's be honest here, testing with the Rasa chatbot was not as easy as the others. It was so much more satisfying though - there's something about the more primitive look of the chat dialog that gave us multiple "Eureka!" and "Aha!" moments.

When it came down to testing our work in real-time, Google, IBM, and Amazon's platforms were a lot more intuitive. The three platforms had conversational agents that could be trained and tested after each change or adjustment. All of them looked pretty in their own way, and each one would give us relevant error messages that pointed to where we misconfigured the bot.

IBM Watson had a tiny edge by labeling each response and command with intents and entities. This gave us the knowledge to understand our dialog and how the bot comprehends our commands.

Google Dialogflow and Amazon Lex had more similar experiences as both platforms used a real-time conversation box to show our bot's training progress. The real-time conversation box reflected what the chatbot would look like using their platform, but Amazon Lex did a better job of labeling all variables in the responses. For example, in the image below the intent and slots were labeled to show what the chatbot "understood" out of the conversation or command.



Yes

Thank you. Your Business Projector is reserved for pickup under the name Bill on 2018-10-29. See you soon!

Inspect response

Dialog State: Fulfilled

● Summary ○ Detail

Intent: RentBusinessProjector

Slots	(4/4)
Name	Bill
Phone	6046046004
PickupDate	2018-10-29
ReturnDate	2018-10-30

This was great, but what if we want to share this testing experience with others? We definitely could – in fact you can tinker with our chatbot here.

This is where Google Dialogflow had an edge. If you are a novice chatbot-builder, you are likely to seek feedback from humans that understand these chatbot platforms better. In the case of testing your Google Dialogflow agent with others, it's entirely possible. There is an option to share a link to the chatbot or embed the chatbot so that others may use it or test it immediately.

Each platform had its unique strengths in the testing phase. There was no clear winner here because each platform may have a different appeal to individuals. For example, the IBM interface provides a ton of information we can work with, but the interface might feel less familiar than Google Dialogflow and Amazon Lex.

The testing of each platform would depend on the user's skill level as well. More advanced users would gravitate towards different platforms depending on their needs. For beginners, all that really differed was the look of the testing environment and how the intent or entity information was presented.

Tie between Google Dialogflow, Amazon Lex, and IBM Watson.

Winner Overall

Which platform made it easiest to train and build a conversational agent? It was a very close decision. Although it took more time and resources to train Google Dialogflow, we did end up with a great end result. On the other hand, Amazon Lex provided a sample we could configure right away: a chatbot for a flower shop. This made the configuration process simple and easy to follow along. It took only 2 hours to configure the Lex chatbot before it understood a user's requests and commands related to projector rentals.

Although we did say earlier that IBM Watson took the lead in the configuration experience, the example chatbot we configured was built for "conversations". This is not the ideal use case for Quick N' Easy Projector Rentals when compared to Amazon Lex's flower shop example.

With Google Dialogflow, it took 3 hours to configure the chatbot because there was no prebuilt example that could be applied to Quick N' Easy Projector Rentals. This meant investing more time in the configuration process to define our own intents and entities. Sparking our creativity and building our own intents and entities was all part of the configuration process. Although it was nice to have examples to play around with like Amazon Lex and IBM Watson did, the learning experience Dialogflow provided was invaluable.

For Quick N' Easy Projector Rentals, *the ease of configuration was better* with Amazon Lex because it was **faster** and **easier** to get started by using the example e-commerce chatbot.

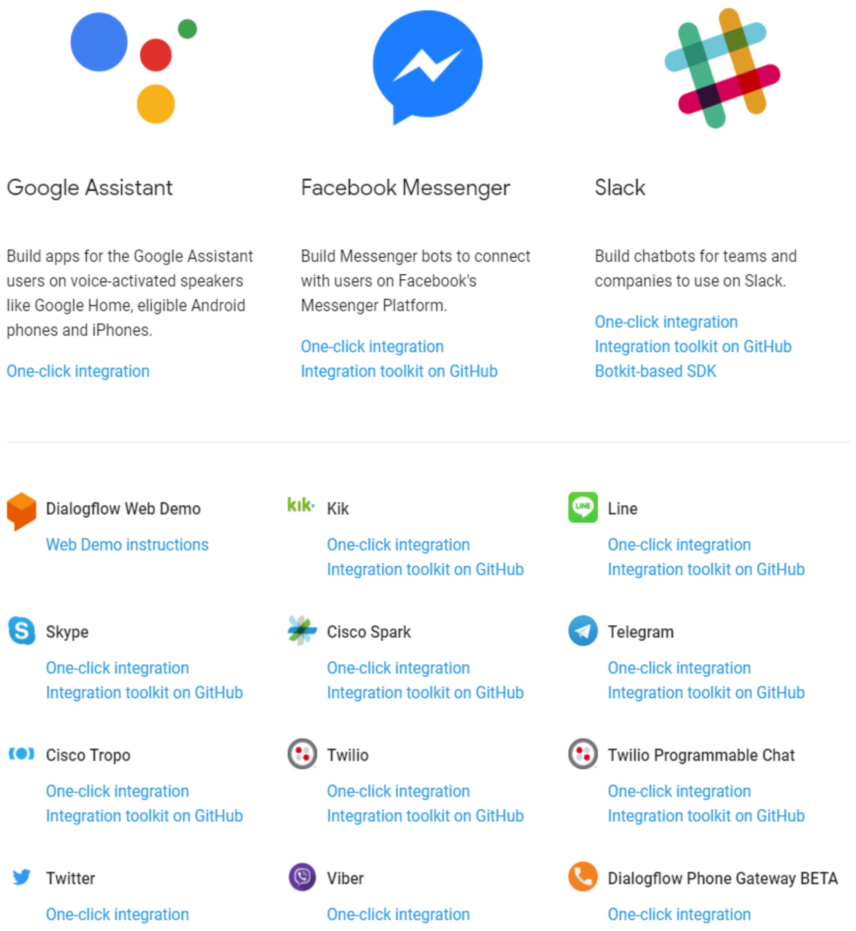
We also looked at our results from a business perspective. Amazon Lex would make fulfilling these orders simple. This is reflective to Amazon's entire business model - using this platform means integrating a chatbot that can smoothen out or improve our projector rental fulfillment process.

Getting the input information of these "rentals" would require Amazon Lex to invoke the *Lambda function code hook* to fulfill the user's intent. This is a more complex feature that this article does not cover.

On a basic level, the AWS Lambda function links to Amazon Lex so that it can push the user's intents to a fulfillment application. For example, if a user confirms they would like to rent a projector, then the Lambda function validates the request and marks it off as an order.

Google provides a similar function through Google Firebase and the Google App Engine. Google Dialogflow can use a custom webhook, Firebase webhook, Google App Engine, and interestingly enough the Amazon Lambda to fulfill user requests. All you need to do to set up a fulfillment function is to enable the webhook call for the specific intent you want to be fulfilled. Once you have flipped the "on" switch, you are given an example fulfillment code that can be customized.

Google Dialogflow seemed the easiest to integrate and put into action. Dialogflow has the most compatible conversational platforms among the pack. Each platform can make use of the Google Dialogflow agent with the click of one button. Much like Amazon Lex, Google Dialogflow can also create a webhook request to fulfill our projector rentals. This fulfillment process can be integrated into any of the supported platforms.



Google Assistant

Build apps for the Google Assistant users on voice-activated speakers like Google Home, eligible Android phones and iPhones.

One-click integration

Facebook Messenger

Build Messenger bots to connect with users on Facebook's Messenger Platform.

One-click integration
Integration toolkit on GitHub

Slack

Build chatbots for teams and companies to use on Slack.

One-click integration
Integration toolkit on GitHub
Botkit-based SDK

Dialogflow Web Demo
Web Demo instructions

Kik
One-click integration
Integration toolkit on GitHub

Line
One-click integration
Integration toolkit on GitHub

Skype
One-click integration
Integration toolkit on GitHub

Cisco Spark
One-click integration
Integration toolkit on GitHub

Telegram
One-click integration
Integration toolkit on GitHub

Cisco Tropo
One-click integration
Integration toolkit on GitHub

Twilio
One-click integration
Integration toolkit on GitHub

Twilio Programmable Chat
One-click integration
Integration toolkit on GitHub

Twitter
One-click integration

Viber
One-click integration

Dialogflow Phone Gateway BETA
One-click integration

Because of our overall experience with building a complete and fully functional agent for Quick N' Easy Projector Rentals, **we crowned Amazon Lex as our winner**. Did we factor in user error? There are a few things that you might have noticed during this experimental project. Intents and entities may have been configured inefficiently. Example utterances and prompts may have been too excessive or not thorough enough. There are certainly areas where human error might have skewed the results, but this is all a part of the overall experience.

WHAT WE LEARNED

What We Learned About Configuring a Chatbot

Someone without any prior hands-on experience in coding, chatbots, and machine learning can still build conversational agents with a little time investment. The chatbot in our example may not have been developed to work in real life scenarios since it only responds to a single case, but they could be built further to work in real situations and use cases.

For the theoretical business *Quick n' Easy Projector Rentals*, building a chatbot was surprisingly simple. The most user-friendly experience had to go to Google's Dialogflow. As opposed to other chatbot platforms, Dialogflow made the training experience easy to manage.

Now that we've picked Amazon Lex as a winner, does that mean it is the go-to platform? For our use-case, it worked with little effort.

If you are going to run through an experiment like this, then you're probably going to have a different experience building chatbots.

This experiment taught us that the most powerful chatbot platform in our case is the one that successfully satisfied our unique use case with as little effort as possible. Amazon Lex gave us the prebuilt intents, entities, and toolset to successfully produce a working chatbot for Quick N' Easy Projector Rentals.

For more advanced users, the power of a conversational agent might be measured by its machine learning algorithm, framework, user interface, or integration capabilities.

Each chatbot platform had its own advantages and drawbacks. What was most important to us was how these chatbots handled user intents, entities, and other variables. What we learned was that it wasn't the chatbot platform or knowledge in coding that limited our results. It was the lack of user intents, entities, and other variables that can be fed to the platform's algorithms so that it could learn how to communicate better.

For our use case, we didn't have a ton of user intents, entities, and example utterances to make the chatbots communicate fluidly. Take our Google Dialogflow agent for example. If we spent more time configuring each chatbot platform, then the result for each would have been a lot more fleshed out.

Do You Need a Specialized Dataset for Your Chatbot?

It's a race to the top, and all competitors are fueled by datasets accumulated through real human interactions. The way we communicate and behave are traits that we want machines to understand and learn. In the race to creating a machine learning algorithm that powers the most sophisticated A.I., companies like Amazon, IBM, and Google are at the forefront of data collection. The best way to teach chatbots how to communicate and interact with humans is to provide it with real examples. Real examples in the form of digestible data.

Keep in mind that there must be a balance between high-quality data and the quantity of data. In our case, additional intents and entities certainly would have resulted in a more sophisticated chatbot. The more thorough these intents and entities are, the more likely the chatbots can understand natural commands and conversations

Whether it's a conversational chatbot or autonomous technology, the machine learning algorithms are fueled by the data you feed to them.

Want to talk about custom data collection?

Book a Free Consultation

Learn More About Custom Data Collection