Leveraging Serverless Technology for Chatbots

Justin Zwick - jaz2130
Sam Kececi - slk2172

Introduction: Motivation

Partly due to the widespread adoption of mobile phones and other smart devices, there has been a precipitous increase in the number of software chatbots. Chatbots are even starting to appear as the primary service in products like Apple's Siri, Microsoft Cortana, and Amazon Alexa.

A chatbot intuitively seems like an excellent candidate for a serverless implementation due to:

1) the high amount of idle time of the application
2) the existence of a clearly defined 'trigger' (the sending of a message into Facebook Messenger)
3) The existence of a plethora of third-party vendors providing free Natural Language Understanding services to developers such as Lex (eliminating much of the required logic that would otherwise need to be implemented in a long-running server).

In this analysis, we will implement two chatbots -- on the same Facebook Messenger platform -- with identical conversation features, using both a serverless model and a traditional monolithic cloud-based architecture to determine if that intuition is correct.

For serverless, we will be using AWS Lambda because it is the most widespread serverless tool available currently, so we felt it would be the best benchmark for this technology. Likewise, for the cloud server model, we used Heroku, which is one of the most widely used cloud platforms today. Both chatbots were developed using python to ensure uniform development processes and results.

Innovation and Intended Audience (What is new?)

Up until now, there has never been a comprehensive study into the advantages and disadvantages of using serverless technology for a chatbot. Our study will provide insight into which system provides a better option to create a chatbot, serverless or a traditional server model. In other words, this project will analyze whether serverless technology is better suited for chatbots.

Intuitively, it would make sense that serverless would be perfect for chatbots due to the high amount of idle time of the application, the existence of a clearly defined 'trigger' (the sending of a message into Messenger), and a plethora of third-party vendors providing free NLP chat services to developers. This study will examine if reality actually reflects intuition.
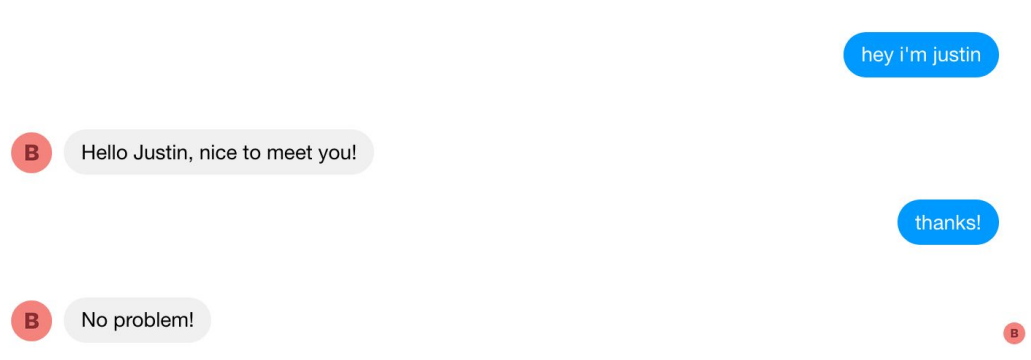
This projects provides immense value to the software engineering community seeking to design chatbots. For nearly every engineer, there are several key metrics which must be researched when deciding on a technology. Chief among these are ease of development, cost effectiveness, and performance (in terms of both speed and storage space needed).

Functionality

Here, we will outline the features of the bots. Both of our chatbots have identical functionality, but will not necessarily respond using the exact same phrasing since the underlying Natural Language Understanding tools they are using are different (Lex for the

serverless bot, and DialogFlow for the server bot). Lex is a NLP tool provided by Amazon in the AWS ecosystem, making it very easy to integrate with AWS Lambda. We felt that to get a proper feel for the development process, we should implement at least 5 different conversation topics. The pictures below demonstrate the functionality of both of our chatbots. For the sake of brevity, we have included just the Lambda chatbot's responses, although both chatbots have the same functionality.
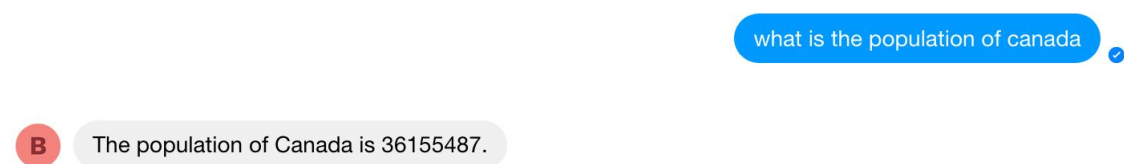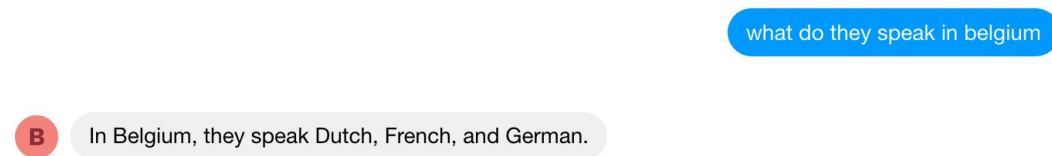
Basic Greetings:

hey i'm justin

B Hello Justin, nice to meet you!

thanks!

B

B No problem!

Currencies:

what is the currency in iceland

B Iceland uses the Icelandic króna.

Populations of Countries:

what is the population of canada

B The population of Canada is 36155487.

Languages of Countries:

what do they speak in belgium

B In Belgium, they speak Dutch, French, and German.

Current Weather Forecasts of any City:

what is the weather like ✓

B Okay, for what city?

san francisco ✓

B In San Francisco, it is 69.22 degrees with Clouds

Capitals of any Country:

what is the capital of iraq ✓

B The capital of Iraq is Baghdad.

Development Process

Serverless Chatbot:

- Lex allows you to directly associate intents with corresponding Lambda functions within the AWS console, making adding new intents trivial
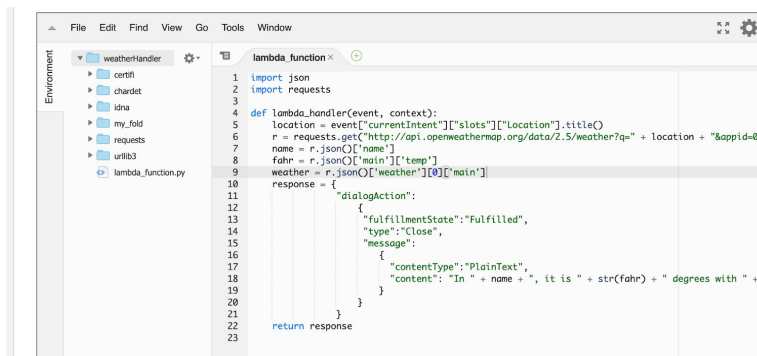
  ▾ Fulfillment ⓘ
    ● AWS Lambda function   ○ Return parameters to client

    Lambda function currencyHandler ▾
                         View in Lambda console ⬀

    Version or alias Latest ▾

-
- To implement the code for each Lambda function, you simply type the code within your browser in the AWS Lambda console, pictured below

```
File  Edit  Find  View  Go  Tools  Window

weatherHandler
  certifi
  chardet
  idna
  my_fold
  requests
  urllib3
  lambda_function.py

lambda_function ×

1   import json
2   import requests
3
4   def lambda_handler(event, context):
5       location = event["currentIntent"]["slots"]["Location"].title()
6       r = requests.get("http://api.openweathermap.org/data/2.5/weather?q=" + location + "&appid=04
7       name = r.json()['name']
8       fahr = r.json()['main']['temp']
9       weather = r.json()['weather'][0]['main']
10      response = {
11          "dialogAction":
12              {
13                  "fulfillmentState":"Fulfilled",
14                  "type":"Close",
15                  "message":
16                      {
17                          "contentType":"PlainText",
18                          "content": "In " + name + ", it is " + str(fahr) + " degrees with " +
19                      }
20              }
21          }
22      return response
23
```

-

- The workflow to add a new intent is quite straightforward. First add a new intent within the Lex console. Then, type "utterances" -- example questions that mimic how a user might want to trigger this intent. Then, associate that intent with a Lambda function, which will be run once that intent is triggered. A picture of utterances for the weather intent are pictured below:

-

Cloud Server Chatbot:
The development process of the Heroku chatbot was a bit less straightforward the the Lambda version. Because there is no built in NLP like Lex, DialogFlow (a google service) was used. Although we initially planned to use Lex in both, we discovered that using Lex with Heroku would be incredibly difficult, due to it being tailored to the AWS ecosystem. Instead, we decided to use DialogFlow, which provides identical NLP services. The main time was spent writing python code that was needed to translate the user's message into a response. Additionally, the DialogFlow NLP must be configured to match the behavior of the chatbot. Below is shown the console of DialogFlow for the languages intent.

Setting up the Heroku server involves essentially the same process as pushing code to Github, making this stage of the process fairly easy. Simply commit code and then push to the Heroku server.
The most challenging part of using the Heroku server involves communicating with Facebook Messenger. Unlike Lex, which easily communicates with Messenger, all of the Heroku communication must be coded from scratch. Some skeleton code was taken from an open

source repository (see appendix) to provide this functionality with some necessary modifications.

Finally, in comparison to the AWS Lambda server, a much larger amount of code needs to be written. While Lex handles most of the communication between Messenger and also triggering the API calls, all of this python code needs to be written from scratch in the Heroku process. One great thing that came out of this development process is the fact that the Heroku code (specifically the chatbot.py file on the github repo) can be used as a skeleton code by anyone looking to make a chatbot on Heroku. To our best knowledge, there does not exist any other open source codebase that offers the same capabilities. Since we plan to make all of our code open-source, it will offer engineers a great starting point to making a chatbot.

Architecture

Serverless Chatbot:
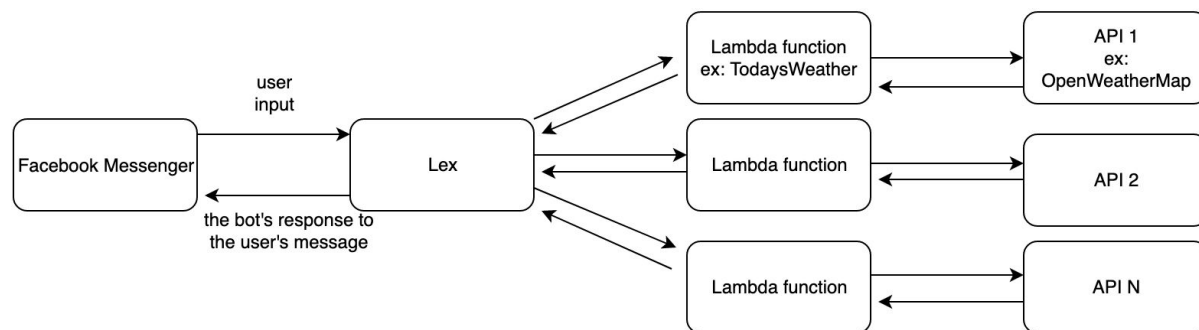
The text that the user sends to the Facebook Messenger bot is forwarded to Lex via a webhook. Once Lex receives the message, it uses Amazon's proprietary NLP technology to match the message with an intent. An intent is a common concept within the chatbot community present within almost all NLU tools (including Lex and DialogFlow), and it roughly corresponds to a conversation topic or a goal the bot's user wants to achieve (ex: booking an appointment or finding out the weather).

Within the Lex console, you can associate each intent with a corresponding Lambda function, such that when a message is received and classified under that specific intent, it's corresponding Lambda function is called. For instance, in the picture below we can see that when a message is received that Lex classifies as the Currencies intent, the currencyHandler Lambda function is then called.

Fulfillment ⓘ

◉ AWS Lambda function   ◯ Return parameters to client

Lambda function   `currencyHandler` ▾

View in Lambda console ⎘
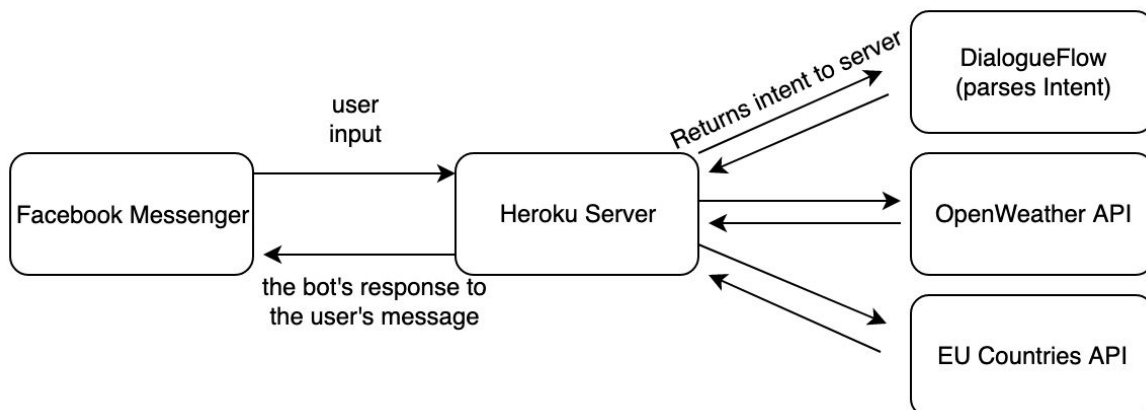
Version or alias   `Latest` ▾

The Lambda functions are used to make API calls that are necessary to fulfill each intent. For example, the languagesHandler Lambda function is used to make calls to the RESTCountries API, as pictured below.

Cloud Server Chatbot:

For the Heroku chatbot, the process of parsing and responding to a user's message is as follows. As the diagram shows, first the Heroku server processes the message, which it sends to DialogFlow to get the intent. DialogFlow then responds with the intent of the message (say it responds with "weather" for example). Then the Heroku server then makes the corresponding API call (finding the weather in this case). Finally, the server constructs the message and responds to the user.

Qualitative Research Questions

R1) Boilerplate Code: Since both chatbots were developed separately by Sam and Justin, we can provide a good look into the challenges that developers face when using each respective technology. *As a developer, is it easier/faster to do the initial setup for a serverless chatbot, or a traditional chatbot?*

R2) Ease of Adding Intents/Different NLP Systems: Intents roughly correspond to a conversation topic or a goal the bot's user wants to achieve (ex: booking an appointment or finding out the weather). *As a developer, is it easier to add new intents using a serverless model or using a traditional model?*

Quantitative Research Questions

R3) Performance and Speed: AWS Lambda and Heroku are built on very different platforms, meaning they can vary greatly in performance and computation time. *Which chatbot performs better in terms of speed and resource usage? Which technology provides a faster response time for messages?*

R4) Financial Considerations: Given the number of messages that large scale chatbots can potentially receive, companies and organizations will likely want to keep their costs low. *Which hosting option is more financially prudent given large scale usage?*

Analysis: Answering the Research Questions

R1) We discussed in detail the development processes of each chatbot. After analyzing these processes, we have concluded that the AWS Lambda process is slightly easier for several key reasons. First of all, unlike the DialogFlow and Heroku, both Lex and Lambda fall in the AWS ecosystem, making connecting the two services relatively seamless. Additionally, Lex communicates directly with Facebook Messenger without the need for code, while Heroku requires extensive code to be written in order to send and receive messages through the messenger API. Finally, AWS Lambda does not require a persistent HTTP server to be running, while the Heroku chatbot needed a Flask web server to be configured and set up to run on the cloud.

R2) Adding intents in DialogFlow and Lex are both equally easy and can be done from within the DialogFlow web console or the Lex web console with the press of a button. However, implementing the actual intent is much easier using Lex. Because Lex is within the Amazon/AWS ecosystem, you can make an intent directly trigger a corresponding Lambda function to accomplish the task. However, in the cloud server bot, the process was much more convoluted. You need to actually manually add code to check what intent was triggered by parsing the JSON response that was received by the server from DialogFlow, and then route it to the correct API call all within the same file.

R3) To measure speed, we can look at the average time it takes to respond to a user's message. While this response time can rely on many things outside of our control – such as the internet connection of the user and the time it takes for the API to respond – most of these factors are constant among the two chatbots. Both chatbots take less than one second to respond to messages and are therefore indistinguishable. This fact is consistent when sending any type of message. However, for the server-based chatbot, there is one factor that takes much longer, and that is when sending the first message after the server is dormant for a long time. Unlike Lambda, Heroku servers (to save resources) will go dormant after inactivity. Thus, the first message sent after a while takes about 20 seconds, which is a very long time to wait.
R4) For both Lambda and Heroku, we paid nothing as for our scale, the services are offered for free (first 400,000 GB-seconds of compute time per month are free in AWS, and low traffic servers are free for Heroku). Using data from a study conducted by Michael Rockford [1], our analysis shows that for the chatbot use-case AWS Lambda is a more cost-effective solution when considering the higher scaling.

What We Learned

Justin Zwick: Prior to this project, I had never built a chatbot before. It was interesting learning about the basic components of chatbots such as intents and utterances, as well as getting hands-on experience using the Facebook Developer Tools to integrate it with Facebook Messenger. In addition, I've never made HTTP requests using Python before, so it was cool learning how to do that.
Sam Kececi: I learned a great deal about the development processes. In addition to having a great understanding of how to build a chatbot, I also learned about developing a web app on Heroku, which is a really important skill to learn in any software engineering job. Finally I learned about important NLP techniques which can be applicable to a wide range of applications.
We both learned that the new serverless technology offered by platforms such as AWS Lambda are a promising improvement in many areas over more traditional cloud platforms like Heroku.

Division of Work

Justin created the AWS Lambda chatbot and Sam created the Heroku chatbot. By separately creating the chatbots, it allowed us to accurately assess the ease of development from a organic perspective. We both worked together to develop metrics and compile the report.

Documentation/Citations

All of the source code that we wrote is available in our GitHub repository. The README.md file contains detailed instructions for using our code to set up your own chatbot.

The code for both chatbots (and documentation to set up the project) can be found at:
https://github.com/zwickers/chatbots

Part of the following repository was used to handle the "under-the-hood" message sending to Facebook messenger [2]

[1]
https://medium.com/@GoRadialspark/heroku-alternatives-aws-azure-and-google-cloud-platform-870ae316527e
[2] https://github.com/codecraf8/facebook-messenger-bot-python-flask