University Chatbot

Tanvi Pandey (tpandey@scu.edu)
Surya Kiran Udaya Kumar (sudayakumar@scu.edu)
Ashish Kumar (akumar10@scu.edu)
Pragya Kathpalia (pkathpalia@scu.edu)

Github link:
https://github.com/tanvi05pandey/CloudComputing

***Abstract*—The use of chatbots in different spheres of modern life is widely common. They help us make bookings for travel, order things online, provide financial assistance for simple transactions, and many more. Despite such common prevalence, we saw a need for a chatbot in our university's admissions office. The purpose of our chatbot is to help students get answers to their questions instantaneously when applying to the university. It also helps lessen the burden of people working in the admissions office.**

Our chatbot is powered by Amazon Lex and has a conversational interface on Slack. Once deployed on the university's website, it is easily scalable and can handle large amounts of queries at the same time.

In its current state, our chatbot is easy to use, intuitive, and very efficient. Students no longer need to sift through the university's website to find information. They can simply shoot a query to the bot and get answers instantaneously. It saves time for both the students and the people working in the admissions office.

Keywords—Chatbot, Amazon Lex, Amazon Lambda, Amazon S3, Slack, Intents, Lambda Function

I. INTRODUCTION

Many of us have been part of the admissions process when applying to a university. As an incoming student, we have lots of questions - application process, deadlines, required documents, visa process and many more. Most of the time these questions are answered by people in the admissions office. Now imagine if you are in a different time zone than the university's and are forced to wait until the next day because when you sent an email to the admissions office, they were already closed. Wouldn't it be wonderful if there was someone present 24x7 to answer your questions? Now let us shift our perspective to the point of view of people working in the admissions office. Apart from their daily work, they have to answer the same repetitive questions from new students every quarter. How great would it be if someone could take over these mundane tasks from them and reduce their workload? Some might argue that there is information available on the university website. While that is true, there is also a downside to it. If you are applying to many universities at the same time, all the information on their website can be daunting. It takes time to get used to one website and find information that you are interested in and you might not always have the luxury of time. We want something that answers our questions instantaneously, is available 24x7, and can help us from getting lost with the information overload on the website.

There is one solution to all these problems - Chatbot. We have created a chatbot for the university admissions office that can efficiently answer student inquiries. Built using Amazon Lex, (the same conversational engine that powers Amazon Alexa) our chatbot is ready to be deployed on any university's website.

Having experienced firsthand the problem of communication gap during our application process due to different time zones, building a chatbot to solve that problem seemed like an obvious choice. We took inspiration from multiple chatbots

available in the market for all sorts of purposes and came up with one tailored solution for university students.

II. Background and Related work

Artificial intelligence (AI) conversational chatbots have gained popularity over time, and have been widely used in the fields of e-commerce, online banking, and digital healthcare and well-being, among others. The technology has the potential to provide personalised service to a range of consumers. However, the use of chatbots within educational settings is still limited. Through our work, we have demonstrated the use of chatbot as an administrative tool, but the possibilities are endless.

We started out by searching for chatbots already in use by other universities. We found inspiration in one used by Oklahoma State University. This chatbot prompts the user to enter short phrases or keywords for better results. We implemented a similar approach in our chatbot and took it a step further where the chatbot suggests the user specific keywords when it doesn't understand the user's input. It understands the context of the conversation and provides hints to the user accordingly. Our chatbot does not need correctly spelled words to answer a user's questions. It is smart enough to ascertain what the user meant (as long as the spelling is not completely off) in order to return a relevant response. We have harnessed the power of Natural Language Processing, Deep Learning to build a smart, intelligent bot.

III. Approach

The system is made up of three primary components, connected to one another serially. They are:

1. Chatbot interface powered by AWS Lex
2. Fulfillment functions powered by AWS Lambda
3. Data storage and retrieval using Amazon S3

Chatbot interface powered by AWS Lex : Lex is a service for building conversational interfaces for applications using voice and text. It uses the same conversational engine that powers Amazon Alexa. It has a few main components namely, a Bot which performs automated tasks, an Intent that represents an action that the user wants to perform, and Slots, for data that might be required by intents. At runtime, Lex prompts the user for specific slot values.

We had to first define the various queries a student might have, and then depending on the query, we came up with sequences of possible phrases that they might say to the bot. We defined intents, slots and input prompts based on what the student is looking for.

Fulfillment functions powered by AWS Lambda : Lambda is a serverless compute service. It can be configured using functions which are invoked by a trigger, a Lex bot in our case. It features pre-built integration with Lex. The user's messages to the bot invoke lambda functions which actually fulfill the request. Lex uses NLP to automatically identify the lambda function to call based on the user's message. We wrote a total of five Lambda functions in Python, one for each of the main intents.

Data storage and retrieval using Amazon S3 : S3 is an object storage service. It can be tied to the Lambda service using the AWS SDK for Python (Boto3) which provides a Python API for AWS infrastructure services. We created a small database of several files, each containing information relevant to a university, such as application deadlines depending on the major, and uploaded it to a storage bucket using S3. We then connected it to the lambda services, so that the stored information can be fetched upon request.

How it works : The student interacts with the bot UI and depending on their message(s), Lex decides what intent is to be invoked. It invokes the Lambda function tied to that intent and performs the requested action. If the student requests information that is stored in the S3 bucket, the lambda service fetches it and relays it to the user through Lex. If the bot can't confidently decide what the user is trying to say, it invokes the fall-back intent, which provides the user with some default options, and tips to make the message more understandable.

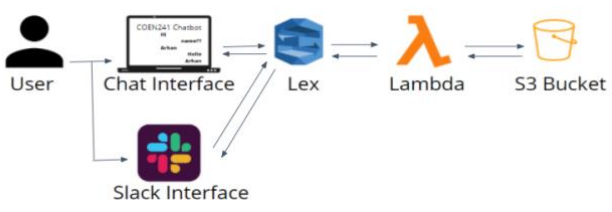

Fig. 1. A high level view of the system

As an augmentation to the bot, we integrated it with Slack, making it accessible to anyone through the Slack app or website. In the course of a day—or a single notification—University might need to cycle among Slack, email, text messages, chat rooms, phone calls, video conversations and the AWS console. Synthesizing the data from all those different sources isn't just hard work; it's inefficient. When something does require your attention, Slack plus AWS Chatbot helps you move work forward more efficiently.

IV. Outcome

After having satisfactorily built and integrated the primary components, we ran it through a series of tests. We took special care for some particular cases, for instance, if the user enters wrong input, the bot reiterates the question. If it is unable to understand the user after repeated attempts, it invokes the fall back intent and the conversation starts over again. For all of the intents we had defined, it worked smoothly. The replies from the bot were almost instantaneous, except when information had to be fetched from the S3 bucket, in which case it took a couple of seconds.

After that, we integrated it with Slack and each of us ran the same tests on there using our individual Slack accounts. It worked perfectly as expected and there was no difference in the results from earlier. Anybody with the link to the bot can converse with it through the Slack website or app. The bot is ready to be deployed on a university website as well.

V. Analysis and Future Work

After a series of chats between the client and the chatbot, if the chatbot isn't able to comprehend the client, the chatbot provides a predefined path that is rigid to the intents developed so far and does not offer solutions to complex queries/ questionnaires. Eg: suppose the client wants to request a change of department while the admission application is already submitted. The chatbot will redirect again to the admissions page rather than tailor a specific answer to the client based on the scenario. The application chatbot is specific to university applications for new students. The SCU's websites and applications can be used to leverage the existing chatbot to serve all the current student needs along with the application-specific FAQs.

In order to cater to specific client needs, a real person needs to be intimated. In such scenarios, adding an SNS(simple notification service) will automatically inform an info desk person about the chat that does not have the solutions which the client expects. Also, the SNS can be configured to inform the person through either SMS or email for real-time assistance. After rigorous testing and checks, the chatbot can be deployed and integrated with the current SCU's website, lowering the need for human assistance and decreasing the wait time of the current chat system drastically.

VI. Conclusion

Summarize the main findings of your project, and what you have learned. Highlight your achievements, and note the primary limitations of your work.

A fully functional chatbot was developed integrating AWS Lex, Lambda, S3 along with Slack. It was interesting to experiment with different cloud technologies and tweak the resources which were specific to certain regions. The integration of S3 buckets with the Chabot was tricky, due to the way the data would be retrieved from the bucket. It was intriguing to experiment with the session attributes which were shared with different intents, the way lex was structured to sequentially get all the required information and use it intelligently to provide multifaceted solutions was an eye-opener.

The functional chatbot was integrated with the Slack messaging platform, providing a clean and user-friendly interface that communicated directly with the Lex bot. All the cloud resources had to be synchronized and free of any latency so that the dialog flow with the chatbot would be in near real-time. Although the chatbot would be a breeze to set up and integrate once the development stage is completed, the bot would need constant support so that it is current and serves the latest information from the databases and would need updates/patches that improvise the chatbot after analyzing the user trends whilst using the chatbot.

REFERENCES

[1] https://aws.amazon.com/lex/

[2] https://aws.amazon.com/lambda/

[3] https://aws.amazon.com/s3/

[4] https://docs.aws.amazon.com/lexv2/latest/dg/what-is.html

[5] https://medium.com/bukalapak-data/one-easy-way-to-develop-your-own-chatbot-dacdb231957c

[6] https://www.ijert.org/dynamic-conversational-chatbot-using-amazon-web-services

[7] https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/oracle-s3-integration.html

[8] https://docs.aws.amazon.com/lex/latest/dg/gs-bp-create-bot.html

[9] https://www.velotio.com/engineering-blog/serverless-chatbot-with-amazon-lex

[10] https://aws.amazon.com/blogs/machine-learning/building-a-real-time-conversational-analytics-platform-for-amazon-lex-bots/

[11] https://aws.amazon.com/blogs/machine-learning/greetings-visitor-engage-your-web-users-with-amazon-lex/

[12] https://aws.amazon.com/premiumsupport/knowledge-center/lambda-execution-role-s3-bucket/

[13] https://docs.aws.amazon.com/lex/latest/dg/slack-bot-assoc-create-assoc.html

[14] https://medium.com/@rashimparmar/building-chatbot-using-amazon-lex-amazon-lambda-and-amazon-s3-96b11c37f5e2

[15] https://aws.amazon.com/getting-started/hands-on/create-banking-bot-on-amazon-lex-v2-console/?trk=gs_card

[16] https://docs.aws.amazon.com/lexv2/latest/dg/deploy-slack.html

[17] https://slack.com/blog/collaboration/aws-chatbot-bring-aws-into-your-slack-channel