Problem Solving Chatbot for Data Structures

Ankil Shah, Bhargav Jain, Bhavin Agrawal, Saurabh Jain, Simon Shim
Computer Engineering Department
San Jose State University,
One Washington Square, San Jose, California – 95192-0180, USA
ankil.shah@sjsu.edu; bhargav.jain @sjsu.edu; bhavin.agrawal @sjsu.edu; saurabh.jain @sjsu.edu; simon.shim@sjsu.edu

***Abstract*—Intelligent chatbot, is a system which can interact with humans and answers questions on a certain domain. Today, the challenge is to build a system which will resemble human brain. Generally, the brain stores the memory in a decentralized manner across the brain with the help of neuron as opposed to a centralized manner in computer file system. There are short term and long-term memory storage with different priority based on variety of situation. The system can take inputs in written or voice format and respond the question from a knowledge base. In most cases a chat bot does not have problem solving capabilities. Our system can solve data structure problems using deep neural network (DNN). With a given dataset the system can provide services to access data in format such as arrays, stacks, queues and trees. Based on these data structures we can solve problems like traversing lists, reversing numbers and translating language of syntactic divergences. The learning service is not an algorithmic program rather a trained model using DNN. With the implementation of problem solving chatbot, it will understand how to organize and retrieve data based on user's data structure choice. We used Neural Stack Machine (NSM) with Recurrent Neural Network (RNN) as the controller.**

Keywords—Chatbot, Alexa, (RNN)Recurrent Neural Network, (NSM)Neural Stack Machine.

I. INTRODUCTION

Chabot's are the systems which are designed to simulate conversation with human. One of the main important functions of Artificial Intelligence is to develop the system to find solutions to the problems. Chat bots are the next step in the internet evolution.

In the early 21st century, with the expansion of globalization internet has become the most convenient way to share information. It provides the platform that allows users to participate, offer feedback, share ideas and receive information. Moreover, it is important to manage these resources wisely, relevant communications, timely response to the questions on the problems like data structure questions and question-answering (QA) from conversation.

The problem-solving Chabot can provide very convenient way for users to communicate via Echo platform (e.g. Amazon Echo). An intelligent assistant can simulate human like speech conversation. It will allow user to submit a question or sentences into an audio input format and in return, intelligent assistant can generate a response which can be spoken by means of a voice synthesis such as Amazon Echo..

Building an intelligent Chabot which can adept to human brain like capabilities of memorizing, accepting user's preferences and providing relevant response can be vital for data structure problems. Human brain stores the information in differently designated areas such as long-term memory and short-term memory in a decentralized manner with the help of the connected neurons. This is different compared to traditional file system which stores information in a centralized manner. An intelligent assistant that can resemble features of brain is capable to provide the best results for complex tasks like solving data structure problems. The assistant can remember important information in long term storage based on criteria such as information repetition, importance of situation or some predefined conditions. Trivial information can be stored in short term storage in order to erase in future.

In this paper, we aim to use Artificial Neural Networks which employs Machine learning and Deep learning algorithms for memorizing and retrieving the processed information. In contrast to conventional neural networks, Recurrent Neural Network (RNN) has the ability to read and write Neural Stack Machine (NSM) same as conventional computer does [1]. Recurrent neural network (RNN) is capable to solve synthetic questions pertaining reasoning and interference in natural language, when trained with supervised learning. RNN uses three attention mechanisms to provide the best recommendations and output to the user [1].

In addition to, we are using recurrent neural network with Neural Stack Machine (NSM) for sequential processing of dataset. The recurrent neural network has been proved effective for language understanding and question answering problem [3]. As described in [4], building Chabot is relatively easy with recurrent neural network. We aim to combine it with the Neural Stack Machine (NSM) as depicted in [16] to get the best results for data structure problems. In this project, the workflow of the system starts with the Echo platform. We have used Amazon Echo for this purpose. The Amazon Echo features a personal assistant called Alexa that accepts the audio input from the user. Alexa uses the services of the Alexa Voice Service (AVS) which adds intelligent voice control to Amazon Echo. Alexa Voice Service receives the voice recording as an input and convert it to text response with the use of Natural Language Processing (NLP). This text input will be provided to RNN. RNN process the received input and retrieve relevant information from the Neural Stack Machine. Eventually, it generates a text response as an output. The text response can be provided as a voice output by following the path through AVS and eventually Amazon Echo. RNN can

also store the important information (e.g. contained in the input) to the Neural Stack Machine.

This system has the ability to solve complex tasks which require the Neural Stack Machine read and write.

A. *Proposed Approach*

Our project is based on neural networks where we aim to develop a problem-solving chat bot. Neural network is adept at the task of classification like image recognition and speech recognition but that does not hold true for the tasks which includes arithmetic computing and logic computations.

It has been shown that neural network failed to compute even addition of two binary numbers. This approach addresses the solution where neural network can learn those arithmetic and logic operations which can be trained by end to end backpropagation. [7] Neural Programmer can combine the steps to produce output and call augmented operations over several steps.

We believe our work through the project would provide us with deep insight in the domains of deep learning, machine learning and the implementation of RNN and Alexa. Moreover, the integration of DeepCoder, a system which can write program itself makes the chatbot smart enough for the programming world [8][13].

B. *Current State of the Art*

Problem solving chat bot on amazon echo platform will change how user communicates with electronic platforms tremendously. Various developments are going on in this field to implement neural network which can help in problem solving and interact with amazon echo platform.

Generally, decision tree algorithm is used for storing the data. IBM Watson has the capability of Natural Language Processing (NLP) which analyze the text and classify the data and form an Ontology. There is a Relationship Extraction tool which extracts relationship from the given data and tries to classify the objects with appropriate attributes. The extraction service uses the classification, association and segmentation.

In classification model by using the decision tree, regression, neural networks techniques it tries to predict the output fields based on input fields.

Another approach is inducing latent programs with gradient descent. Neural network is adept at the task of classification like image recognition and speech recognition but that does not hold true for the tasks which includes arithmetic computing and logic computations. It has been shown that neural network failed to compute even addition of two binary numbers. This approach addresses the solution where neural network can learn those arithmetic and logic operations which can be trained by end to end backpropagation. [7] Neural Programmer can combine the steps to produce output and call augmented operations over several steps. [11]

These both of the approaches address to several problems but does not help much to remember the past history of events which can help to add intelligence to neural network. If RNN is combined with Neural Stack Machine [16] where neural network can write data and read data from, we can remember the past actions for longer period of time and prompt user with relevant information. RNN uses different attention mechanisms to store data into neural network and read as per requirements. [1] RNN has greater capacity to solve complex and structured task which could have been provided without Neural Stack Machine read write. RNN when combined with Neural Stack Machine performed better than long short-term memory (LSTM) and neural turning machine to answer synthetic questions.

II. PROJECT ARCHITECTURE

A. *Introduction*

High level architecture of our project is shown above – Problem Solving Chat Bot for Data Structures. This architecture is comprising of 3 tiers.

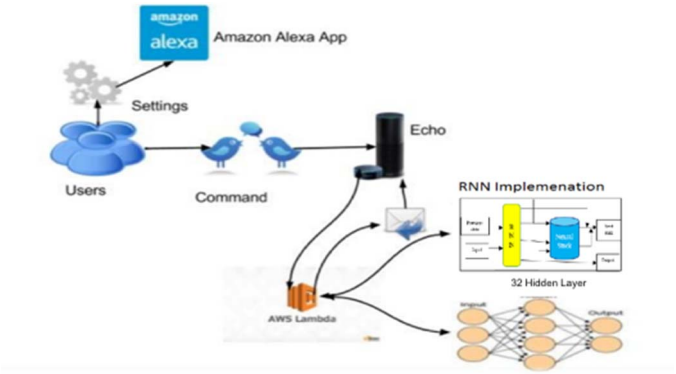

Fig. 1. Architecture Diagram of Personal Assistant for Problem Solving Chat bot for Data Structures.

The 1st tier is presentation tier which comprises of Amazon Alexa mobile application and Amazon Echo hardware which is Amazon's question answer bot.

The 2nd tier is interaction tier which comes between backend neural network implementation and Amazon's echo hardware. This tier comprises of NLP processing 16 system, AWS Lambda or HTTP end points.

The 3rd tier is the heart of the application. This tier will have Artificial Neural Network implementation using RNN approach or it will use Facebook's memory networks for backend processing. RNN will be implemented using Tensor Flow or open source NTM-Lasagne library by Snips [12].

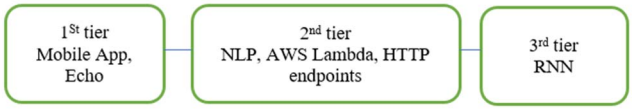

Fig. 2. 3 tier Architecture Diagram of Problem Solving Chat bot.

Our paper aims to customize Amazon echo in such a way that it could serve as problem solving chat bot for data structures like Stack, Queue, Linkedlist, Binary Tree. For example, reverse the stack, use stack to implement queues, perform traversals for binary tree. Also, other examples for problem solving chatbot can be meeting reminder with session maintained and inference question answers from set of information. We are aiming towards making Alexa an intelligent response agent which can directly communicate to its users. The project would have learning capabilities which can be useful in above mentioned areas. Below are the set of questions in sequence that user can ask.

Alexa, can you add number to the stack?

Alexa, reverse the numbers in the stack?

Can you add one more?

Alexa, can you again reverse the numbers from the stack?

Currently Alexa has limitation that session is not maintained between sequence of the questions. For example, if first question is "Alexa, is there a meeting tomorrow?" and next question is "Where"? Alexa cannot infer that user is asking for the location of the same meeting which has been asked initially. Alexa is only able to answer question which is asked one time. Thus, there is no inference reasoning available in Alexa. So, we are improving system by implementing session using neural networks.

The project would have storing capabilities of previous results with the use of RNN with Neural Stack Machine, which can lead Amazon echo towards an intelligent assistant agent. It will use feedback from the past results or past queries. In that way, echo would be able to answer continuous/connected questions. This type of environment will be very useful in creating better user experience.

B. *Architecture Subsystems*

The combination of customized Alexa mobile application and Amazon echo hardware which can be used to enable/disable our developed service chatbot.

Interaction between hardware and the backend will be handled by Amazon Alexa's Natural Language Processing system and AWS Lambda service. NLP system will convert commands from the users to raw texts or logical texts and gives input to AWS Lambda service for further logical processing. There will be two-way communication between AWS lambda service and backend.

The 3rd and most important component is RNN with Neural Stack Machine (NSM) which will have capabilities of self- learning and self-adapting.

The detailed architectural components of the paper are:

User: The user is the consumer which can interact with alexa and ask to perform different operations on the data structures. He is also the consumer of this system who can adjust the settings of Amazon Alexa mobile application and Amazon echo hardware as well he can give commands to the system to perform desired action or expects intelligent response.

Amazon Alexa mobile application: This is the mobile application which is used to connect and enable different service settings of Amazon Echo hardware. Here, the application will be customized and will have one additional service named chatbot data structures.

Amazon Echo hardware: This is Amazon's Question/Answer bot which will be customized to be an intelligent response agent for data structure problems. Hardware will accept the commands from the users and delivers these commands to backend processing system. In response to it will accept answer from backend system and deliver answer in the form of voice to user.

Commands: Commands are user speech which will include voice sentences like perform a traversal for a given list.

AWS Lambda: AWS lambda is a service which is useful for the logical processing of the texts and for routing new information. It has different functions which can be useful in inferring knowledge.

III. PROJECT IMPLEMENTATION

A. *Client Implementation*

In our client side implementation, the client can interact with Amazon Echo by saying "Alexa problem solving chatbot for data structures". Once the Alexa skill is loaded into the Alexa, the client can interact with the Amazon Echo. He can train Alexa first by prompting which data structure to use to perform operations like stack traversal or reversing string using a particular data structure. Another example is to generate a Linkedlist and reverse it. All these data are being taken by the Alexa and the neural network is trained to provide a user with the best possible result. The data set is first being trained for the given numbers using neural stack and RNN and stored in the data store. Later a user can prompt the given list of numbers and Alexa would response by learning from the trained network.

For example, to generate a linkedlist and reverse it . All these data are being taken by the Alexa and the neural network is trained to provide a user with the best possible result. The data set is first being trained for the given numbers using neural stack and RNN and stored in the data store. Later a user can prompt the given list of numbers and Alexa would response by learning from the trained network.

B. *Middle-Tier Implementation*

Middle tier includes connection logic as well as business logic. There are two approaches that we have discussed to deploy our business logic in AWS Lambda.

- Use AWS Lambda as API server: In this approach, We will create python API server using AWS Lambda and deploy another Linux server on AWS. We will use that server to perform training and testing neural networks. We can also expose https endpoints from API server. Advantage of this approach is that, entire dedicate server will perform only training and testing of neural network. That speeds up the process. But drawback of this approach is unnecessary resource utilization as here we are using AWS Lambda as well as another dedicated server. That sums up to extra cost to the system.
- Use AWS Lambda as API server as well as for training and testing: We can combine everything into one and deploy as a single python service on AWS Lambda. As AWS Lambda is a server less computing, we don't have to manage underlying resources which are needed. We also don't have to worry about allotting CPU memory, storage and also configuring Python based environment.

We have used second approach. In combination with AWS Lambda we are using CloudWatch for debugging and testing. CloudWatch specifies resources that we have used and also, we are using CloudWatch logging as a debugging tool.

We are using RNN with NSM where RNN will learn to how to use the stack to implement an algorithm. It will learn when to push and pop data as per model and training given to that Neural Network.

C. *Neural Stack Machine*

Neural stack behaves as simple stack but RNN will use to push and pop with the use of backpropagation. This stack can learn how to take sequence of inputs, memorize the given sequence and convert it to appropriate required pattern.

In the learning phase RNN will control NSM and will decide when to push and pop data accordingly. This push-pop sequence will predict output. Output will be compared to given input and error will be calculated. We are using backpropagation to update the weights of hidden layers of neural network which will help to predict output correctly.

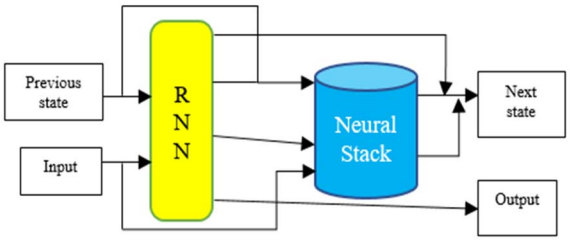

Fig. 3. Neural Stack Machine controlled by RNN.

In the RNN we are using hidden layer recurrence strategy of RNN where last hidden layer is dependent on each previous input layer and previous hidden layers. There are instances where neural network can break when 2 inputs are same in given inputs.

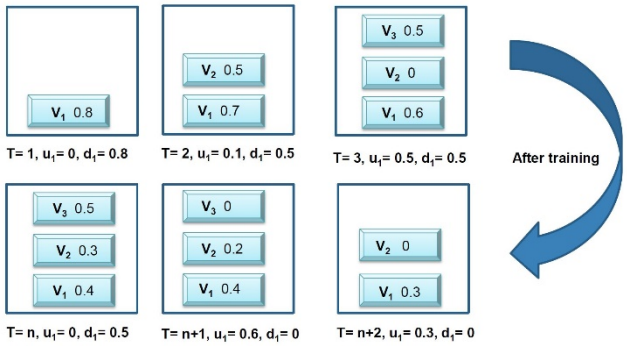

Fig. 4. Neural Stack Operation.

Here, as shown in diagram, vectors V1, V2 and V3 have to be pushed on the neural stack. The value in the right side of the vector is weight of that particular vector. We can also have weight as a 0 for some of the vectors. (Special condition when we need to pop particular vector). Character t shows the timestamp, character u shows the pop weight for particular timestamp whereas character d shows the push weight for the certain timestamp.

Now, we will understand figure for each timestamp in detail. At timestamp t = 1, we have pushed the vector v1 with the push weight 0.8. In timestamp t = 2, we have push weight 0.5 and pop weight 0.1. so, weight for vector v1 changes from 0.8 to 0.7 whereas new weight is inserted for vector v2 as 0.5. In timestamp t = 3, we have both push weight 0.5 and pop weight also 0.5. So, we will be inserting new vector V3 with weight 0.5 and now because we have a pop weight 0.5, we will have to reduce this 0.5 from V2 and V1. We will first reduce weight from V2 and after reducing all weight from V2 if we have still some weight to be reduced, we will reduce from V1. So, we will reduce 0.5 weight from V2's weight. Consequently, we will have V2's weight 5 -5 = 0. This is the initial condition where neural stack is learning to replicate the behavior of the stack. Once we have trained the network enough, pop weight will be subtracted from the top vector as shown in the image and consequently vector will be popped in the reverse order of they were pushed.

$V_t[i]$ can be represented as value of V at previous time stamp t-1, where i is between 1 to timestamp t. Also, $V_t[i]$ is same as v_t when i and t are the same [16]. V_t is the state of stack's memory at time t, so stack might have multiple vector at particular timestamp. So, V_t will have list of list of V vectors.

$S_t[i]$ can be calculated by subtracting total strength for all vectors from pop wait when, i is between 1 and t. It is the same as push wait d_t when i = t. Here, S_t will represent the list of vectors for strength of vectors at different timestamp. For example, at t = 1, there will be a strength for vector V1 only but at timestamp t = 2, there will be a strength of 2 vectors named V1 and V2.

In learning phase, RNN will control Neural stack machine and will decide when to push and pop data accordingly. This push-pop sequence will predict output. Output will be compared to given input and error will be calculated. We are using backpropagation to update the weights of hidden layers of neural network which will help to predict output correctly.

We are using hidden layer recurrence strategy of recurrent neural network where last hidden layer is dependent on each previous input layer and previous hidden layers. There are instances where neural network can break when 2 inputs are same in given inputs.

D. Data-Tier Implementation

This tier comprises of NLP processing system, AWS Lambda or HTTP end points. It will have Artificial Neural Network implementation using Recurrent Neural Network (RNN) approach using Neural Stack Machine. RNN is implemented using Tensor Flow and open source python NumPy library. RNN learns when to push and pop on NSM. On each iteration, it updates the model by updating the push weights d_t, pop weights u_t, strength s_t, stack states v_t. This model will be stored in file system which can be used later to predict the correct output.

E. Technology Descriptions

We have used various technologies for the project. Below is the list of technologies used:

- Python
- AWS lambda
- RNN
- NumPy or TensorFlow
- Neural Stack
- Alexa Skill Kit

IV. Result and Analysis

We have trained our model with different approaches and measured accuracy for the same. We change different parameters which can affect the training of the model. For an example, we used different batch sizes and number of training examples. We trained model with batch size of 10 and traning examples 200000, 400000 and 800000 and the same number of traning examples with batch size of 50.

These results were obtained while training the model with variation of number of training examples and batch size. The comparison of accuracy estimation with the number of training set are shown in figure 5. Also, we did a comparision for 16, 32, 48, 64 layers. The best output was observerd after 99999 iterations with 64 layers with 99.53% accuracy. Moreover, it was observerd that with 64 layers, when we trained the model with more that 100000 training set the accuracy started decreasing because of the overfitting problem. Hence, best results were obsesrved with 99999 interation for Stack related data structure problems.

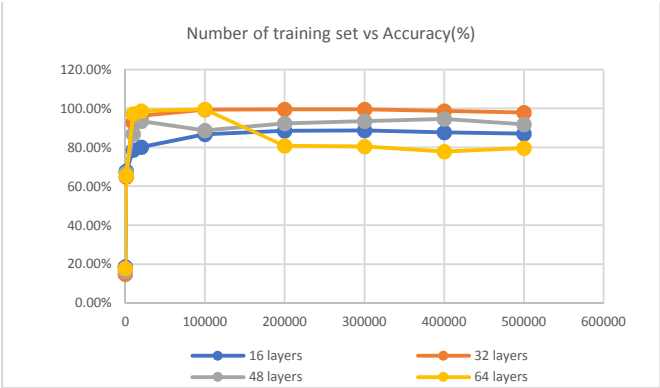

Fig. 5. Experiment 1: Number of training set vs Accuracy(%) for Stack

V. Conclusions And Future Work

To conclude, we believe that with the integration of neural network, machine learning algorithms with interaction model like Alexa we can ease complex problems.

We have created an intelligent Chatbot to interact with a user to make it behave in a human like manner. Just like how humans interact with humans and remember their preference we have stimulated it using NSM. The neural network is first trained to remember different data structures. The weights for different parameters are set when we train the network and these weights are stored in the database. Then later when a user request for different operations for different data structueres, the weights are determined for different operations and the results are being retrieved.

We believe that the approach and the research study which we have used in our paper has the ability to solve complex tasks. The neural network and machine learning algorithms which are used for the data structure problem can be extended to other problems such as to solve the Deep Coder [8] problem which can write the code based on how the network is being trained, finding shortest path for traversal for graphs or Subway traversal problem [1] for finding the best possible path based on user preference.

References

[1] Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., Grabska- Barwińska, A., ... & Badia, A. P. (2016). Hybrid computing using a neural network with dynamic external memory. Nature, 538(7626), 471-476.

[2] Recurrent neural network. (2017, Feb 21). Retrieved Feb 22, 2017, from https://en.wikipedia.org/wiki/Recurrent_neural_network.

[3] Peng, B., & Yao, K. (2015). Recurrent neural networks with external memory for language understanding. arXiv preprint arXiv:1506.00195.

[4] Kovalevskyi, V. (2016, December 11). Own ChatBot Based on Recurrent Neural Network. Retrieved February 22, 2017, from https://blog.kovalevskyi.com/rnn-based- chatbot-for-6-hours-b847d2d92c43#.zi22qcx5z.

[5] Weston, J., Chopra, S., & Bordes, A. (2014). Memory networks. arXiv preprint arXiv:1410.3916.

[6] Sukhbaatar, S., Weston, J., & Fergus, R. (2015). End-to-end memory networks. In Advances in neural information processing systems (pp. 2440-2448).

[7] Neelakantan, A., Le, Q. V., & Sutskever, I. (2015). Neural programmer: Inducing latent programs with gradient descent.

[8] Balog, M., Gaunt, A. L., Brockschmidt, M., Nowozin, S., & Tarlow, D. (2016). DeepCoder: Learning to Write Programs. arXiv preprint arXiv:1611.01989. https://arxiv.org/abs/1611.01989.

[9] Di Martino, B., Esposito, A., D'Angelo, S., Marrazzo, A., & Capasso, A. (2016, July). Automatic production of an ontology with NLP: Comparison between a Prolog based approach and a Cloud approach based on Bluemix Watson service. In Complex, Intelligent, and Software Intensive Systems (CISIS), 2016 10th International Conference on (pp. 537-542). IEEE.

[10] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).

[11] Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. Proceedings of the IEEE, 78(10), 1550-1560.

[12] T. (2017, March 21). Neural Turing Machines library in Theano with Lasagne. Retrieved March 27, 2017, from https://github.com/snipsco/ntm-lasagne.

[13] Di Martino, B., Esposito, A., D'Angelo, S., Marrazzo, A., & Capasso, A. (2016, July). Automatic production of an ontology with NLP: Comparison between a Prolog based approach and a Cloud approach based on Bluemix Watson service. In Complex, Intelligent, and Software Intensive Systems (CISIS), 2016 10th International Conference on (pp. 537-542). IEEE.

[14] Weston, J., Chopra, S., & Bordes, A. (2014). Memory networks. arXiv preprint arXiv:1410.3916.

[15] Sukhbaatar, S., Weston, J., & Fergus, R. (2015). End-to-end memory networks. In Advances in neural information processing systems (pp. 2440-2448).

[16] Grefenstette, E., Hermann, K. M., Suleyman, M., & Blunsom, P. (2015). Learning to transduce with unbounded memory. In Advances in Neural Information Processing Systems (pp. 1828-1836).

"How to Code and Understand DeepMind's Neural Stack Machine," How to Code and Understand DeepMind's Neural Stack Machine - i am trask. [Online]. Available: https://iamtrask.github.io/2016/02/25/deepminds-neural-stack-machine/.