Intelligent Travel Chatbot for Predictive Recommendation in Echo Platform

Ashay Argal[1], Siddharth Gupta[1], Ajay Modi[1], Pratik Pandey[1], Simon Shim[1], Chang Choo[2]
Computer Engineering Department[1] and Electrical Engineering Department[2]
San Jose State University,
One Washington Square, San Jose, California – 95192-0080, USA
ashay.argal@sjsu.edu; siddhart.gupta@sjsu.edu; ajay.modi@sjsu.edu; pratik.pandey@sjsu.edu; simon.shim@sjsu.edu; chang.choo@sjsu.edu

***Abstract*— Chatbot is a computer application that interacts with users using natural language in a similar way to imitate a human travel agent. A successful implementation of a chatbot system can analyze user preferences and predict collective intelligence. In most cases, it can provide better user-centric recommendations. Hence, the chatbot is becoming an integral part of the future consumer services. This paper is an implementation of an intelligent chatbot system in travel domain on Echo platform which would gather user preferences and model collective user knowledge base and recommend using the Restricted Boltzmann Machine (RBM) with Collaborative Filtering. With this chatbot based on DNN, we can improve human to machine interaction in the travel domain.**

Keywords*— *Chatbot, DNN (Deep Neural Network), Restricted Boltzmann Machine (RBM), Amazon Alexa Echo platform, Natural Language Processing (NLP), Machine Learning (ML), Neural Net(NN).

I. Introduction

To design and implement natural and intuitive interaction modalities is a primary research field in the Human-Computer Interaction Domain. Systems that can interact with user in their natural language are being researched vigorously at present. Voice enabled chatbots are becoming more and more popular with the advent of devices and technologies like Google Home, Amazon Echo, NLP, ML, AI etc. Chatbot is an artificial service that can start, continue and handle complex interactions with human partners in their natural language. Voice enabled chatbots, today, are considered as classical yet innovative interfaces for natural language interaction with machines [1].

The Chatbot simulates a real-world travel agent that achieves some result by conversing with the machine in a dialogic fashion using natural language. The forefront of voice-driven digital assistants is from the big four: Amazon's Alexa, Google's new assistant, Apple's Siri and Microsoft's Cortana. The MIT Technology Review lists conversational interfaces as one of the ten breakthrough technologies of 2016 [2].

Currently text based chatbots are used widely in commercial web and entertainment systems. What we propose is a voice based chatbot which interacts with user in their natural language through speech for enabling users to search for queries regarding travel domain. Successful implementation of our system will understand what you are saying, analyze it, and give you a suitable response if the query is related to the travel domain.

The user interacts with Amazon Echo device which triggers the Alexa skill that processes the request and constructs a suitable response from the data retrieved from the knowledge base. The request is processed by the backend server which is hosted by Amazon Web Services (AWS). The back-end database is incorporated into a knowledge base which has been trained using user ratings. The user query is analyzed, and the response is generated using the recommendations from our deep neural network (DNN) engine which implements Restricted Boltzmann Machine for collaborative filtering [5] based on RBM used for count data [7].

The need for intelligent agents is growing with the widespread use of personal machines and the desire of their makers to provide natural language interfaces. The proposed system will change how to recommend travel itinerary by incorporating personal preferences, remembering its users' intentions and travel history, and generating responses using collaborative filtering.

II. Background

Today, the traditional travel agents present many issues that can be conquered by the Intelligent one. The traditional ones are subject to their availability, the Intelligent Agent is available throughout. Moreover, no agent in this world can "know it all". The data present with an actual travel agent can be limited or outdated, this makes it not so efficient. In addition to that, when a customer calls a travel agent after some months, most of them do not remember the history and the interests of the caller.

The Amazon Echo platform, used to design this Chatbot, is gaining its popularity exponentially. A new report from Consumer Intelligence Research Partners (CIRP) estimates that there are now 8.2 million customers who own an Amazon Echo device [3]. The Alexa voice service provided by Echo platform is powered by Amazon's cloud-based artificial intelligence which can answer questions, control smart home devices, play games with users, set timers, and much more.

To predict user intention and update the memory, a learning based system usually learns from many hand labeled data points. This is a major barrier when launching a dialogue system to a

978-1-5386-4649-6/18/$31.00 ©2018 IEEE

new domain/system, because much efforts are needed to either collect hundreds of thousands of hand-labeled training data for machine learning algorithm to learn or writing many rules takes too much time. The system should be able to adapt to a new domain or business with decent performance [4].

III. Chatbot Architecture:

Our System takes input in the form of user speech through Amazon Alexa enabled device and analyses that input using NLP techniques to find out what the user is trying to say or ask and responds accordingly. As most chatbots our system also consists of three main modules: a knowledge base which is the heart of the chatbot, a speech recognition engine as an interface between users and machine, and an interpreter program. Our Speech recognition engine is the echo device, interpreter program is our Alexa Skill and we have created a knowledge base by data mining and data scraping and bundled it with Machine learning and AI algorithms to provide an intelligent travel Engine. It is built using technologies like MongoDB, MySQL, Elasticsearch, Neural Networks and a successful implementation on Restricted Boltzmann machine for Collaborative filtering.

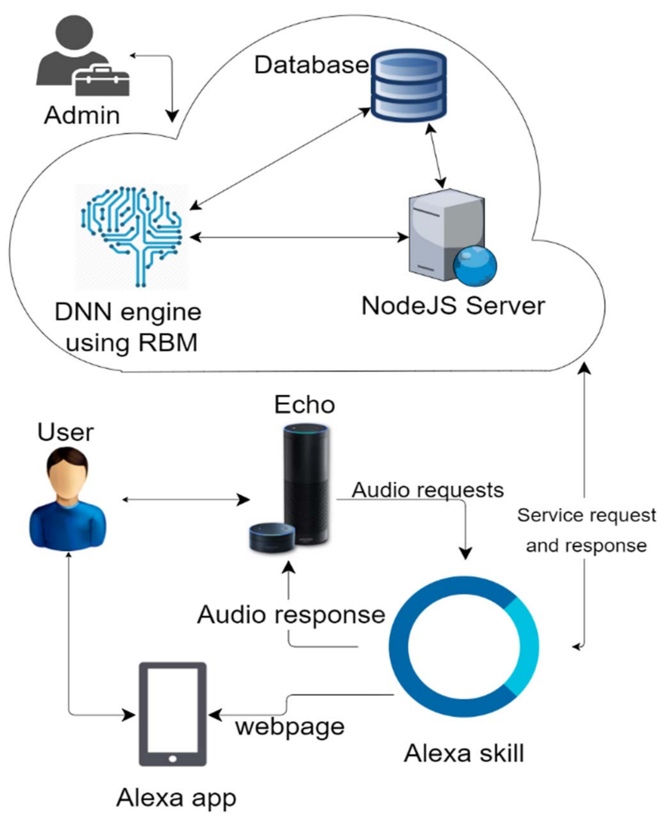

Fig. 1. High-Level Architecture

The front end of the system is an Amazon Echo device and the Amazon Echo mobile application. The user interacts with the Echo device powered by a voice recognition software which converts speech to text and vice versa. The mobile app is used for displaying Echo generated cards and user registration page of the service. The Alexa skill is the interface for communication between the Echo device and the system server. The Alexa skill receives Audio requests from the Echo device and sends the service request to the AWS Lambda function that triggers process to get adequate response of the user query, it is also responsible to call relevant custom APIs that are hosted in EC2 instance. When a suitable response is generated, the Alexa skill generates an audio response and sends it to the Echo device which communicates with the user. The Alexa skill is also responsible for generating cards and webpage. The Echo cards act as a visual front to the user.

The back-end of the system is hosted on the cloud and has a server, Travel AI engine, custom APIs and a database. The handler function is hosted on AWS Lambda which interacts with the custom APIs to generate user-suitable responses. The NodeJS server, hosted on our private server, powers the entire system. Custom APIs are the tools to retrieve relevant information from the web hosted by other service providers.

The hybrid database consists of a MySQL server and a MongoDB server. The MySQL server hosts user sensitive information like user bookings and their financial details and the MongoDB server hosts through REST API data such as the booking objects, user preferences and their recommendations, these contributes to the knowledge base of our system.

IV. User Requirements:

Essential feature of the system is that the Alexa device must be able to handle complex conversation with the users regarding their travel queries. Also, it is important to create and maintain a knowledge base with the help of machine learning and AI algorithms to provide accurate service to the user requests. The knowledge base should provide accurate replies for the user requests; thus, it must be trained properly to predict the reply. A database would be required to store and manage the knowledge base and store user information. Also, it is essential to deploy an Application server which will communicate with the device and database to service user requests. Thus, the essential requirements are as follows:

- Alexa Device
- Application server
- Knowledge Base
- Machine learning Algorithms(Training)
- Correct travel datasets
- Database
- Cloud Hosting
- Web Client

V. Proposed System:

Sometimes the online website or travel agents are unable to provide the adequate and accurate travel information to the user and in such context the intelligent travel chatbot is a key. The intelligent travel chatbot is designed to first take all the necessary inputs from the user to predict the relevant and accurate answer to the query of the user. The system first identifies the missing information and probe the user further to collect this missing information to make the original query which needs to be answered. The original query is answered by taking into consideration the user preferences, the past travel history and the user ratings collectively.

Suppose there are one or more missing data in the user's query; in such case, the bot is designed to ask the question until the answer fulfills the required missing information. After all the information is fulfilled the query is sent as request to the request handler to analyze what the user is saying and activate the proper algorithm to find the appropriate response.

A. System Architecture:

The system is developed and maintained by using Agile methodologies. The system consists of various independent modules integrated together to provide all necessary services. Those modules are as follows:

1) Flight Module: This module will handle all types of requests related to flights, which includes flight searches, booking, schedule etc. It should be able to handle all types of conversations related to flights.

2) Rental Car Module: The car booking module to be designed and determine the pickup location, drop off location, time and the number of guests to book a rental car. The module would help the user to find a car of their liking in an easier manner.

3) Hotel Module: The hotel booking module to be designed and determine the destination, time and the number of guests to book the hotel room for. The module would help the user to find a hotel of their liking in an easier manner.

4) Web Application Module: To make the project more efficient, safe and less time consuming, there is a need to make a web application to get the user personal details, confidential information like credit card details, user preferences in relation with travel in advance which helps user to directly initiate and manage travel related decision based on the recommendation.

5) Recommendation Module: Recommendation Systems are a type of Information Filtering Systems which implements an algorithm to predict the ration that a user would give to a product. In simple terms, Recommendation systems try to predict what user would like to buy/watch/read etc. depending on the use case. The system provides Hybrid recommendation system which utilizes both content based and collaborative filtering algorithm to predict recommendations. This also helps us to tackle the famous cold start problem of Collaborative filtering algorithm. The recommendation module is providing relevant suggestions for Flight, Hotel, Rental Car and User Preferences module.

When the user is new, or we do not have any user activity the recommendations from the content based recommendation are given preference over the recommendations from collaborative filtering, as the user history builds up the recommendation from the collaborative filtering system becomes more accurate and slowly the weights are adjusted to give more preference to recommendations from the collaborative filtering recommendation.

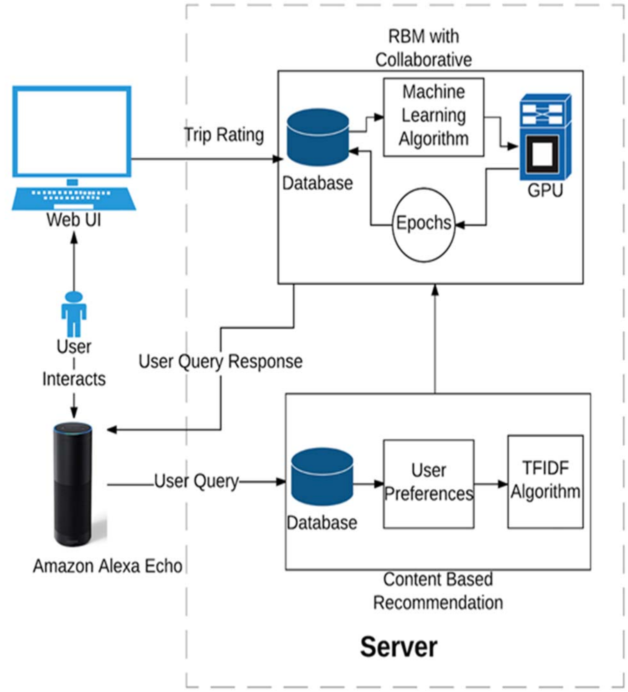

Fig 2. Two-*level* recommendation system architecture

a) Content Based Recommendations: For Content Based Recommendation we have built indexes of all our products which we then compare with the preferences set by users. The indexes are built in Elasticsearch from our Data Corpus. The search is done using Elasticsearch query search which utilizes a combination BM25 and TF-IDF algorithm to find the best recommendations.

b) Collaborative Filtering Recommendation System: Implemented a Restricted Boltzmann machine with collaborative filtering [5] for a Collaborative Filtering Recommendation System. The System first creates a vector for all users and their activity and preferences and then it is used to implement and initialize a neural network. The neural network is used to apply training on large dataset including the data provided by the users. Once the network reaches equilibrium and all the weights acquire their equilibrium values we find the nodes which predict a user recommendation.

VI. Methodology

1) The flight, hotel and car booking module are 3-tier modules hosted in AWS Lambda, custom physical server and our custom DB service. The AWS Lambda is responsible for building up speech, conversating with the user and generating requests by prompting the user to fill the missing spaces. Our Custom server hosts the backend server for the modules with our database

service running alongside and makes API calls to the third-party services and the database. It also hosts our content based recommendation and collaborative filtering modules. MongoDB holds all the user details like preferences and the interests and has a hybrid integration with the MySQL server to hold the user sensitive transactional details.

2) Web app: This module focusses on to provide users the capability to register and login themselves in the application, store all the details in different databases like MongoDB and MySQL. MongoDB is used to store all the user related personal information while MySQL is used to store the transaction and payment related information of users. The module helps to encrypt the confidential data like passwords and credit card related details. This module also provides a profile page for users to see their profile details and users can change the details whenever needed. This module is developed using MEAN Stack

3) Recommendation Module: We gathered all the data available for flight, hotel, and car rental booking using data mining, web scarping and API scraping methods. After having enough data, we stored them in our data model using a default schema which percolates all required information like price, date, type etc. from these gathered objects and stored them in a Database.

After the first-step we had enough data to build our own application and start taking bookings and finally their ratings. We created an Alexa app with a user driven voice user interface to allow users to search for flights, hotels and cars and book them. As in the beginning we did not have enough/any user data a content based recommendation system was used as collaborative filtering was not possible.

a) Content Based Recommendation System:

For this system to work correctly we first created a user profile which contains user preferences like food, car type, car brand, flight carrier preference etc.as given by the user. Also, we indexed the trip objects present in the database to apply fast and robust similarity matching algorithms.

When a search query for a particular trip is done we search for all the necessary criteria like date, city etc. required for a basic search and then score the documents on the basis of matching preferences by first applying a bm25 TF-IDF similarity score and then passing the scored document to a pre-configured elastic script for find most relevant results from our dataset for a particular query.

For our collaborative Filtering to work we provided a feature to rate the bookings they had made on a scale to 1-5 because these are ratings which we use to apply Collaborative Filtering.

b) Collaborative Filtering Recommendation System:

We developed our Collaborative Filtering Recommendation System to work when the dataset for users and ratings is large enough to apply collaborative Filtering. We Knew, Existing Collaborative Filtering approaches fail to handle very large datasets and traditional machine learning methods become pointless.

In this approach, we show how Restricted Boltzmann machines [5] can be used model data like user ratings of different objects like flights, hotels and cars. We focus on the implementation as the comparisons and inferences are provided for this method for a simple one object dataset.

Suppose we have N users, F flights, C cars, H hotels, and integer rating values from 1 to K. We created three different collaborative filtering subsystems for Flight, Hotel and Cars. I will focus on Flights in this paper.

So, we have N users, F flights and integer rating values from 1 to K. To solve the problem of efficiently managing the missing ratings value for each user we created N Restricted Boltzmann Machines, as each user has rated the same set of flights this helps in managing relationship for collaborative filtering.

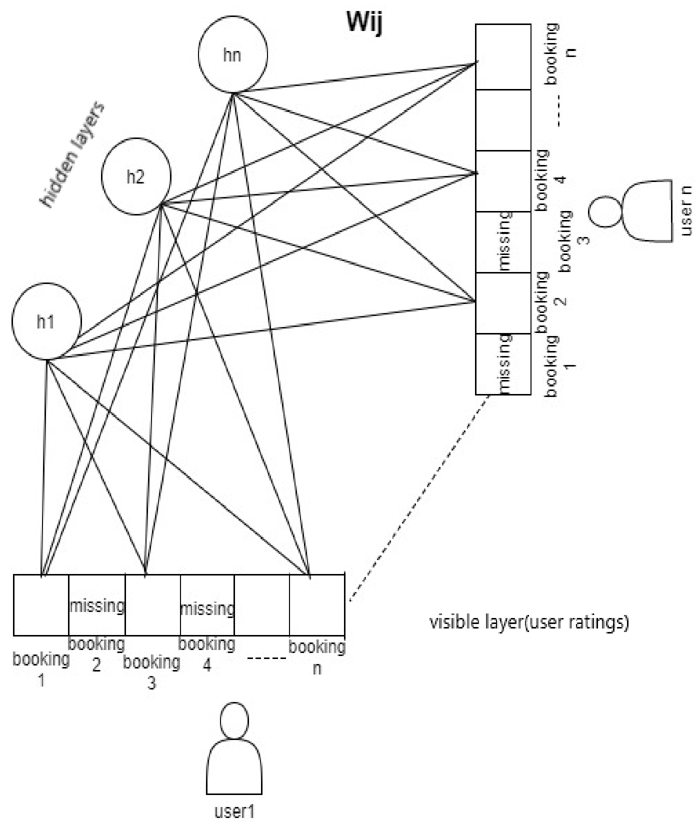

Fig. 3. Hidden and visible nodes in the RBM [5]

Each user is treated as a single training case for an RBM which had M "softmax" visible units symmetrically connected to a set of binary hidden units. Using this config each hidden unit try to learn to model dependencies between ratings for that user. Thus, we use one RBM per user, each RBM has the same number of hidden units but only contains flights rated by user as visible units created using softmax function.

Each user has a different RBM but they all share their weights and biases, thus if two users rated the same flight each of their RBMs much use the same weight and bias to between their own visible softmax unit and the hidden units. The states of all different users can vary a lot, thus for simplicity we will focus on getting gradients only one user with its RBM. Full gradients can be obtained using averaging the gradient over all users, i.e. over all RBMs.

A Softmax model is used for modelling each column of observed visible rating matrix V. To Model the hidden units a Bernoulli Distribution model is used.

Energy of a configuration (pair of Boolean vectors) (v,h)for our RBM system over the visible ratings V is defined as

$$p(V) = \sum_h \frac{exp(-\mathrm{E}(V,h))}{\sum(V',h)\, exp(-\mathrm{E}(V',h'))} \quad (1)$$

The parameter updates are required to perform gradient ascent in the log-likelihood.

To avoid computing model, we follow an approximation to the gradient of a different objective function Restricted Boltzmann Machines for Collaborative Filtering called “Contrastive Divergence” (CD) [6]:

$$\Delta W_{ij}^{k} = \in (< v_i^k h_j >_{data} \; - < v_i^k h_j >_T) \quad (2)$$

It was shown that CD learning is quite efficient and greatly reduces the variance of the estimates used for learning [6]. The learning rule for the biases is just a simplified version of the above equation.

Given the observed ratings V, we can predict a rating for a new query flight q in time linear in the number of hidden units [5].

Suppose that we add w to each of the K weights from the K possible ratings to each hidden feature and we subtract w from the bias of the hidden feature. So long as one of the K ratings is present, this does not have any effect on the behavior of the hidden or visible units because the “softmax” is over-parameterized. If, however, the rating is missing, there is an effect of −w on the total input to the hidden feature. So, by using the over-parametrization of the softmax, the RBM can learn to use missing ratings to influence its hidden features, even though it does not try to reconstruct these missing ratings and it does not perform any computations that scale with the number of missing ratings.

The subtle information about user-rating pairing which cannot be captured by a standard RBM is captured in this method. Our Database tells us in advance which user/flight pairs occur in the test set, so we have a third category: bookings that were done but for which the rating is unknown. This is a valuable source of information about users who occur several times in the test set, especially if they only gave a small number of ratings in the training set.

The conditional RBM model takes this extra information into account. Let r ∈ {0, 1}M be a binary vector of length M (total number of Flights), indicating which flights the user rated (even if these ratings are unknown). The idea is to define a joint distribution over (V, h) conditional on r. In the proposed conditional model, a vector r will affect the states of the hidden units.

Instead of using a conditional RBM, we can impute the missing ratings from the ordinary RBM model. Suppose a user rated a movie t, but his/her rating is missing (i.e. it was provided as a part of the test set). We can initialize ‘vt’ to the base rate of movie t, and compute the gradient of the log-probability of the data with respect to this input (2). The CD learning takes form:

$$\Delta v_k^t = \in (< \sum_j W_t^k h_j \; >_{data} - < \; \sum_j W_t^k h_j \; >_T) \quad (3)$$

After updating v k t, for k = 1 ... K, v k t are renormalized to obtain probability distribution over K values. The imputed values vt will now contribute to the energy term and will affect the states of the hidden units. Imputing missing values by following an approximate gradient of CD works quite well on a small subset of the Netflix data set, but is slow for the complete data set.

VII. DATASET

For our training dataset we created 200 users noting down their preferences for hotel, flight and car rental searches. A sample user object looks like follows:


```
{
 "first_name": "Siddharth",
 "last_name": "Gupta",
 "email": "siddharth.gupta@sjsu.edu",
 "password": "******",
 "gender": "Male",
 "dob": "01-01-1990",
 "contact_information": {
  "mobile": "1234567891",
  "address": {
   "address_line_1": "201South4thStreet",
   "address_line_2": null,
   "city": "SanJose",
   "region": "California",
   "country": "US",
   "postal_code": "95112"
  }
 },
 "preferences": {
  "flight": {
   "airline_name": [
    "emirates",
    "etihad"
   ],
   "airline_class": "EC",
   "airline_days": [
    "Monday"
   ],
   "airline_time": "morning"
  },
  "hotel": {
   "hotel_star_rating": "4",
   "hotel_location": "airport",
   "hotel_price": "50"
  },
  "car": {
   "car_model": "Audi",
   "car_rental_company": [
    "enterprise",
    "budget"
   ],
   "car_mileage": "300",
```

```
    "car_price": "150",
    "car_features": [
      "powersteering",
      "Bluetooth"
    ]
   },
   "food_cuisine": "Indian",
   "food_type": "Vegetarian"
  }
}
```


Here we record user preferences like preferred carrier, preferred hotel rating, hotel location (city or near airport), airline class, car mileage (300mile/day or unlimited), car features etc. This information helps us to build our content based recommendation system.

For our Flight, Hotel and Car Rental Database first we restricted the Database to 50 most populated U.S. cities, i.e. our system can recommend and book hotel, rental cars in 50 cities and flights originating and ending within these 50 cities.

For our flight Database we scarped as much flight data as we could from Google QPX express API for these 50 cities and used it to build our database.

Our Flight Object looks like as follows:


```
{
 "carrier": "Alaska Airlines",
 "flightnumber": 1930,
 "destination": {
   "city": " Los Angeles ",
   "airportCode": " LAX ",
   "airportName": " Los Angeles International "
 },
 "source": {
   "city": " San Francisco ",
   "airportCode": "SFO",
   "airportName": " San Francisco International "
 },
 "price": "126.61",
 "duration": 82,
 "class": " FIRST CLASS ",
 "arrivalTime": "2017-10-10T09:22-07:00",
 "departureTime": "2017-10-10T08:00-07:00",
 "cuisine": [
   "Veg",
   "Continental",
   "Asian"
 ]
}
```


We could not mine Hotel and Car rental data as free API access to such data was not available other than a few requests on Hotwire, thus we mined a few objects from Hotwire and modelled our car and Hotel data according to Hotwire API and QPX API.

Object properties like car/hotel features, hotel/flight cuisine etc. help us to match user preferences. Our Dataset consists of 30,000 flight objects,5000 car objects and 5000 hotel objects.

We made some real users and asked them to book some trips which they like and give dummy ratings to them. We have created a portal through which user can rate his past bookings. These ratings are used to apply collaborative filtering using Restricted Boltzmann Machine Algorithm.

TABLE II COMPLETE DATASET

Data objects	Value
Number of users	200
Number of flights	30,000
Number of cars	5000
Number of Hotels	5000
Number of bookings for car	20,000
Number of bookings for hotel	20,000
Number of bookings for flight	150,000

VIII. RESULTS AND DISCUSSIONS

The result of our project is a product, which fulfills all the goals mentioned above while maintaining a certain quality and efficiency.

A. Execution Environment

- SERVER: DELL™ POWEREDGE™ T110, Ubuntu 14.04 LTS, NodeJS v6, Quad-core Intel® Xeon® 3400 series,32GB RAM
- RBM: GeForce GTX TITAN Z GPU GK110B 12GB, Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz, 32GB RAM, Ubuntu 14.04.5 LTS

The intelligent travel chatbot using Alexa Beta skill builder, provides more interactivity with the user resulting in better conversations. After the implementation, we observe the better interactivity in the following conversation.

User: *Book a hotel for 2 people.*

Bot: *Where do you want to book the hotel in?*

User: *Denver.*

Bot: *When do you want to check-in?*

User: *Tomorrow.*

Bot: *When do you want to check out?*

User: *25th December.*

Bot: *Here are the top results …*

Consider another conversation in which the user gives multiple values in a single thread.

User: *Book a hotel for 2 people in Denver.*

Bot: *When do you want to check-in?*

User: *20 December till 25 December.*

Bot: *Here are the top results …*

After a successful booking the system tries to recommend user according to his current itinerary. The system remembers user slot values for intents using previous booking or conversations in the same session. In the next conversation we show that the system elicits slot values from current hotel booking.

User: *Book a car.*

Bot: *Since you booked a hotel room in Denver. Do you want to book the car in Denver for 2 people on 20th December till 25th December?*

User: *yes.*

Bot: *Here are the top results …*

We are using the slot gathering functionality provided by Amazon Alexa's Beta Skill Builder.

When the user comes back first time after the trip date, Alexa prompts the user to rate their experience. This is shown in the following conversation.

User: *Alexa, Open Travel Chatbot*

Bot: *Hello user, I see that you finished you trip to Denver on 10th November. Would you like to rate your Hotel booking? Please give a rating from 1 to 5 or say "skip".*

User: *4*

Bot: *Thank you for your feedback. You can change your rating anytime by using the Alexa card or the email you received just now.*

This is an important part of our dialog model as the recommendation system is based on these ratings.

The user can also modify his preferences and get his upcoming bookings as shown below.

User: *Show my bookings.*

Bot: *Your upcoming bookings are as follows …. You can now view or edit your preferences, you can now also start new booking.*

User: *Edit my preferences.*

Bot: *Which preferences would you like to edit.*

User: *Hotel room type.*

Bot: *Which hotel room type do you prefer?*

User: *Suite.*

Bot: *Preferences updated.*

The chatbot retrieves the search results based on the user preferences, previous bookings and the similar bookings of other users which come up from our hybrid recommendation system consisting of a content based recommendation engine and an RBM with collaborative filtering recommendation engine.

IX. Performance And Benchmarks

The graph of Error to Epoch is shown in Fig. 4. It contains the error value as we process our data through 3000 Epoch. It clearly shows that the error reduces exponentially as the number Epoch increases.

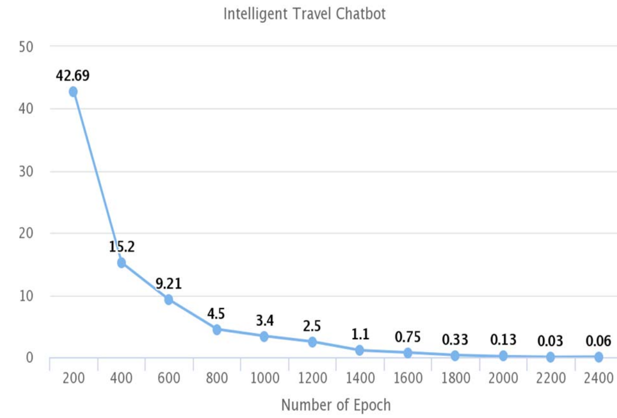

Fig 4. Error vs cumulative *Epoch*

This error in this graph is the mean squared difference between the actual user ratings given and their predicted values from our recommendation system.

Table I shows the system accuracy of RBM as compared to other ML algorithms. This accuracy data was gathered by comparing the ratings given by the user and how often our recommendation system was able to generate the same rating for that particular user and trip.

TABLE III COMPARISON OF VARIOUS ML ALGORITHMS

Algorithm	Accuracy (%)
HMM	70.3
1-FL-SVM	25.9
2-FL-SVM	76.7
RBM	83.2

X. Conclusion

Conversational chatbots will have a significant impact and it would be more beneficial as we gain more personal knowledge about user's preference. Once it is deployed on Alexa Skills market, the application will have an impact on a large number of audiences and will simulate the real-world travel agents.

Implementation of the proposed system shows that most of the basic features available on the travel Websites can be easily incorporated in a voice enabled chatbot. Our system can become a hub of all travel related user queries.

With this implementation we show how a collaborative recommendation system can be applied to service in the travel booking domain to produce more personalized and accurate search predictions for a user while stimulating an intelligent conversation in natural language. The lack of visual interface will be dealt via the use of information cards provided by Alexa and the side web application. The privacy concern must be addressed to safe guard personal information.

XI. REFERENCES

[1] Reshmi, S., & Balakrishnan, K. (2016). Implementation of an inquisitive chatbot for database supported knowledge bases. Sādhanā, 41(10), 1173-1178.

[2] Dale, R. (2016). The return of the chatbots. Natural Language Engineering, 22(5), 811-817. doi: https://doi.org/10.1017/S1351324916000243.

[3] Soper, T. (2017, January 25). Http://www.geekwire.com/2017/8-million-people-amazon-echo-customer-awareness-increases-dramatically/. Retrieved March 7, 2017, from http://www.geekwire.com/2017/8-million-people-amazon-echo-customer-awareness-increases-dramatically.

[4] Sun, Y., Zhang, Y., Chen, Y., & Jin, R. (2016, September). Conversational Recommendation System with Unsupervised Learning. In Proceedings of the 10th ACM Conference on Recommender Systems (pp. 397-398). ACM. doi:10.1145/2959100.2959114.

[5] Salakhutdinov, R., Mnih, A., & Hinton, G. (2007, June). Restricted Boltzmann machines for collaborative filtering. In Proceedings of the 24th international conference on Machine learning (pp. 791-798). ACM

[6] Hinton, G. E. (2002). Training products of experts by minimizing contrastive divergence. Neural Computation, 14, 1711–1800.

[7] Welling, M., Rosen-Zvi, M., & Hinton, G. (2005). Exponential family harmoniums with an application to information retrieval. NIPS 17 (pp. 1481–1488). Cambridge, MA: MIT Press.

[8] S.J. Gong, "A Collaborative Filtering Recommendation Algorithm Based on User Clustering and Item Clustering," Journal of Software, 2010, 5(7), pp. 745-752

[9] J. Wei, J. He, K. Chen, Y. Zhou, and Z. Tang, "Collaborative filtering and deep learning based recommendation system for cold start items," Expert Systems with Applications, vol. 69, pp. 29–39, 2017..

[10] Sedhain, S., Sanner, S., Braziunas, D., Xie, L., & Christensen, J. (2014, October). Social collaborative filtering for cold-start recommendations. In *Proceedings of the 8th ACM Conference on Recommender systems* (pp. 345-348). ACM.

[11] Nilashi, M., Jannach, D., bin Ibrahim, O., & Ithnin, N. (2015). Clustering- and regression-based multi-criteria collaborative filtering with incremental updates. *Information Sciences*, *293*, 235-250.

[12] Nilashi, M., Esfahani, M. D., Roudbaraki, M. Z., Ramayah, T., & Ibrahim, O. (2016). A multi-criteria collaborative filtering recommender system using clustering and regression techniques. *Journal of Soft Computing and Decision Support Systems*, *3*(5), 24-30.

[13] Larochelle, H. and Bengio, Y., 2008. Classification using discriminative restricted Boltzmann machines. In *Proceedings of the 25th international conference on Machine learning* (pp. 536-543). ACM.

[14] Al-Zubaide, H., & Issa, A. A. (2011, November). Ontbot: Ontology based chatbot. In *Innovation in Information & Communication Technology (ISIICT), 2011 Fourth International Symposium on* (pp. 7-12). IEEE. doi: 10.1109/ISIICT.2011.6149594

[15] Fischer, M., & Lam, M. (2016, June). From books to bots: Using medical literature to create a chat bot. In *Proceedings of the First Workshop on IoT-enabled Healthcare and Wellness Technologies and Systems* (pp. 23-28). ACM. doi: 10.1145/2933566.2933573

[16] Goodwin, R., Lee, J., and Stanoi, G., (2005). M.I. Leveraging. Relational Database Systems for Large-Scale Ontology Management in Proceeding of CIDR Conference

[17] Huhns, M. N., & Singh, M. P. (1997). Ontologies for agents. IEEE Internet computing, 1(6), 81-83. doi: 10.1109/4236.643942

[18] Santangelo, A., Augello, A., Sorce, S., Pilato, G., Gentile, A., Genco, A., & Gaglio, S. (2007, November). A virtual shopper customer assistant in pervasive environments. In OTM Confederated International Conferences" On the Move to Meaningful Internet Systems" (pp. 447-456). Springer Berlin Heidelberg. doi: 10.1007/978-3-540-76888-3_67

[19] Shawar, B. A., & Atwell, E. (2007). Chatbots: are they really useful? In LDV Forum (Vol. 22, No. 1, pp. 29-49). retrieved from http://www.jlcl.org/2007_Heft1/Bayan_Abu-Shawar_and_Eric_Atwell.pdf

[20] Vysniauskas, E and Nemuraite, L., (2006). Transforming Ontology Representation From OWL TO Relational Database Information Technology and Control, 35 (3A). retrieved from http://itc.ktu.lt/index.php/ITC/article/view/11779.