IJCSE International Journal of Computer Sciences and Engineering Open Access
Review Paper Vol.-6, Issue-12, Dec 2018 E-ISSN: 2347-2693

Intelligent Travel Bot using Wide and Deep Learning

Siddhant Rele[1*], Danish Ali Furniturewala[2], Sagar Raulo[3], Neepa Shah[4]

[1,2,3,4]Department of Information Technology, Dwarkadas Jivanlal Sanghvi College of Engineering,Mumbai University, Mumbai, India

[*]*Corresponding Author: siddhantrele@gmail.com, Tel.: +91-98201-11435*

Available online at: www.ijcseonline.org

Accepted: 22/12/2018, Published: 31/Dec/2018

***Abstract*—** The usual approach of planning a trip involves a serious of tedious tasks. The manual way is to book through a travel company, which will give you an itinerary for your trip and the costs involved, another way is to book through an online website, where you can pick a place, look for hotel, book a room and then make travel arrangements to your desired destination. The process involved is time consuming and involves looking through various booking websites to find the best bang for your buck. We propose a solution which will make this process as smooth as possible through the use of an interactive travel bot deployed on social media platforms. In this travel bot, a user enters a query asking for a place to stay in a location. The travel bot then constructs a persona based on transactional history of the user, for example, hotels that the user has shown interest in previously. Using this persona and a wide and deep neural network, personalized recommendations are generated by the travel bot.

***Keywords*—** Recommender system, TensorFlow, Wide and Deep Learning, Chatbot, Hotel booking.

I. INTRODUCTION

The traditional approach of planning and booking a trip through a travel company seems to have reached its end with the ubiquitous nature of the World Wide Web. There are a host of travel websites such as Booking.com, TripAdvisor and MakeMyTrip that can be used to plan a trip virtually anywhere in the world. The process of sifting through all of these websites and trying to find the best possible deal is tedious and often time consuming.

To do this job for us, we propose a travel bot which will aggregate all the deals from the top travel websites and provide the best option to the user based on his/her query. Additionally, it will also provide personalized recommendations to the user based on past transactional data.

Why use personalized recommendations instead of traditional ranking based recommendations? You can suggest more relevant travel options to your users when they are customized to fit their profile. Personalized recommendations will make every user feel like your travel bot was curated to provide the perfect recommendations for them. User experience will be significantly improved which increase conversions and click-through rates.

According to a MyBuys[1] study of more than 100 top Internet retailers, personalized recommendations resulted in a 915% increase in the likelihood of the customer buying one of the recommended products.

For example, let us consider the Amazon's recommendations, they appear on the homepage when you login or at the bottom of each page. Amazon provides these recommendations based on the behavior of other Amazon shoppers browsing through various products. In the case of purchasing a portable hard disk drive, Amazon provides a list of similar items other shoppers purchased or looked through while purchasing a hard disk drive.

Our bot will be synonymous to Amazon's concept by utilizing the tendencies portrayed by the customers while booking hotels to predict properties that will stand out and cater to the needs of each customer. This involves the collection of user data on a wide scale and making accurate predictions so that the users feel as if their minds are being read. This can have a huge impact on overall user satisfaction with our travel bot.

The remainder of this paper is organized as follows: Section II provides a review on the existing chatbot architecture and existing implementations in the context of travel planning. Section III provides an overview of wide and deep learning along with our implemented architecture. Section IV provides a summary of the results of our implementation.

II. LITERATURE REVIEW

What is a chatbot?

Chatbots are computer programs that mimic conversation with people using artificial intelligence. They can transform the way you interact with the internet from a series of self-initiated tasks to a quasi-conversation.

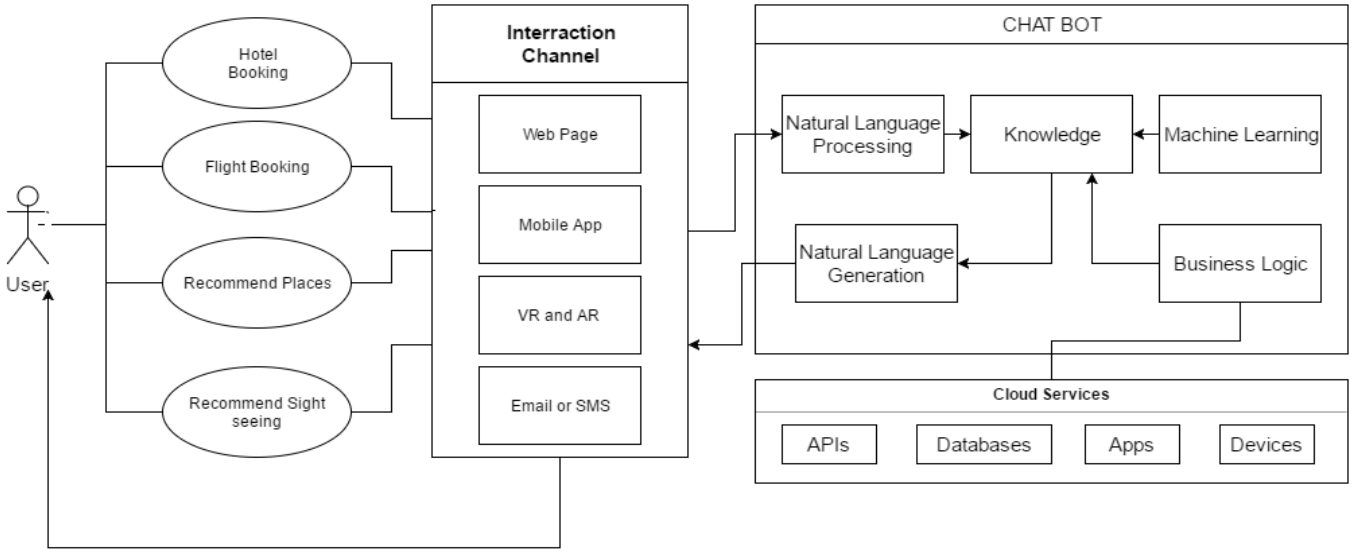

Figure 1: Architecture of existing chatbots

Architecture of existing chatbots

Figure 1 depicts the typical architecture used by the chatbots. It consists of components like Natural Language Processing (NLP) and Machine Learning (ML) integrated with various APIs forming the Knowledge Base. The user typically engages with these bots using a medium such as website, app, email etc. The users' query is then parsed using an NLP engine which extracts intent of the query along with keywords relevant for the business logic to provide results to the query. The business logic then processes the obtained knowledge using various APIs and services to generate results which are then returned to the user using the same medium via which the query was received. We discuss two examples of existing travel bots below:

Marina (Alterra.ai)

Alterra's Marina[2] is a virtual travel agent that can recommend travelers where to go on vacation and sightseeing places, it can also book hotels and flights. It is available on desktop and mobile platforms of Facebook, Slack and Telegram. Alterra can recommend destinations based on activities, interests or themes, it will recommend cities, resorts, national parks, regions or countries to go to. Additionally, it also lets you specify dates, trip duration, weather, etc. "Recommend a place for families", "Recommend a place for adventure", "Please find me 2 roundtrip economy tickets from Chicago to London on Star Alliance next Sunday back on Oct 5 with 1 stop or less", these are some queries you can ask Marina.

Marina uses Neural Networks to power its brilliant Natural Language Processing engine, it makes use of a structured dataset of major vacation destinations across the world, linked with vacation themes, attributes (nightlife, beach, food), possible activities (hiking, fishing), etc. It uses fuzzy search to find vacations suitable for your query. The results are presented as a list of images with suitable subtitled text. The results are sorted according to relevance.

Drawbacks

After asking Marina for a "driveable getaway", from Mumbai, it returned results like Dubai and Sri Lanka, which are not drivable from Mumbai. Additionally, when it could not recognize that New Delhi and Delhi are the same place.

Hello Hipmunk

Hipmunk provides a free personalized travel planner called Hello Hipmunk[3], which simplifies planning a trip through two features: "Hello Email" and "Hello Calendar". Hello Email scans through your emails and searches for references to travel plans, it then finds travel options for the places referred in the email. With Hello Email, you loop hello@hipmunk.com into your email conversations about travel and ask, in the email, something like "Hello @hipmunk, can you send us some flight options for a flight from Boston to Miami from August 8 to 15?" Hipmunk replies, in a conversational, friendly email, with flight options.

Hello Calendar scans a user's Google Calendar to find upcoming travel itineraries and offers a variety of travel options. Additionally, it periodically scans Google Calendar to find events that users are attending in another city. Then, Hipmunk emails flight, train, and hotel or vacation rental search information for those travel dates. It helps you to coordinate travel with friends and family. Hello Hipmunk understands the context of the conversation, just like a real life personal assistant. Travelers can also request to receive price alerts based on their inquiries.

Drawbacks
Full access to Google Calendar and personal emails can raise privacy concerns. Additionally, data provided by these sources is very sparse and hence recommendations are limited in scope.

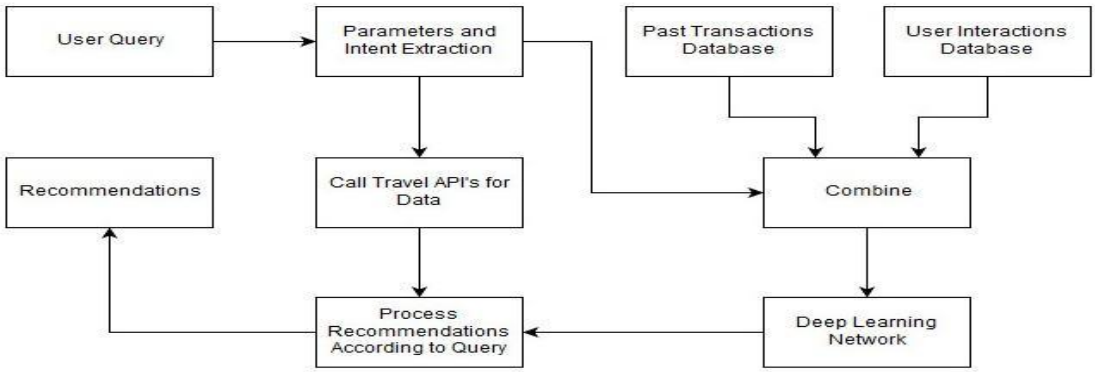

Figure 2: Proposed Architecture

III. PROPOSED ARCHITECTURE

We used the power of deep learning to provide personalized recommendations to a user query related to travel. This system would provide meaningful and accurate recommendations related to a user query on hotels embedded in the flow of a natural conversation interacting with a chatbot. The detailed description of the proposed architecture depicted in Figure 2 is as follows:

Intent extraction
The user query is parsed by the NLP engine (API.ai) which returns a JSON containing the user's preferred parameters and the intent of the user's query, this information is then used to call Travel API's and our Wide and Deep Neural Network (WDNN)[4].

Past transactions
This includes all the transactions (bookings) the user carried out from our platform are stored in a database and used to improve the user's recommendations.

User Interactions
This includes the user indicating whether he liked the hotel recommendations that were generated by our model. They are used to improve quality of recommendations by leveraging the trained model.

Deep Learning Network
The extracted attributes and user's query parameters are fed into our WDNN which generates meaning out of these attributes and provides intelligent recommendations to the user. These recommendations are the passed to the Processing module.

Google Play Store recommendations
Google Play Store uses a WDNN to provide app recommendations to users on the play store, it uses a WDNN trained by a dataset of 500 billion user profile entries. The WDNN used has been open-sourced in the TensorFlow library. The working of the wide and deep models is explained below:

Wide Model
This model is basically a linear model that maps a user query to an item that the user actually wants. This model is used when the user queries something very specific like "J.W Marriott Hotel in Juhu". The output of this query should be the specific hotel requested by the user. The wide model excels at memorizing what people like, but cannot provide an accurate recommendation for a generalized query.

Deep Model
This model is a deep feed-forward neural network that maps a user query to an embedding space with various hotels in them occupying the space. This will help us realize relationships like, people booking a room in "The Taj Mahal Palace" often don't mind staying in "The Trident". However, while populating the embedding space, we find that the deep model generalizes too much, if people want something very specific, it will not give an accurate recommendation.

Combining both models: To overcome the individual drawbacks of wide models and deep models, we combine them to make a wide and deep model which has two distinct parts, a linear model for specific queries and a deep feed-forward network for generalized queries.
As shown in the graph above (figure 3), the sparse features like query="Hotel in Juhu" and item="J.W Marriott Hotel"

are used in both the deep part (right) and wide part (left) of the model. The prediction errors are back propagated to both the wide and deep parts during training. The deep component uses embeddings to find similar items while the wide component can memorize the sparse and specific rules.

Process Recommendations

This module maps the recommendations to the data obtained from Travel API's in order to format the results in the form required by the respective chat platform.

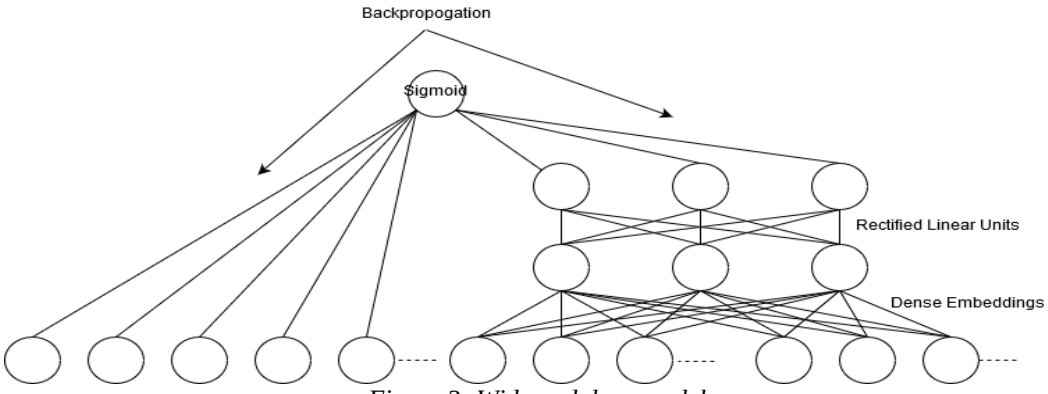

Figure 3: Wide and deep model

IV. RESULTS & DISCUSSION

The WDNN was trained on the review data obtained from Booking.com for 4 cities: Mumbai, Delhi, Paris and New York City.

We have used Rectified Linear Unit (ReLU) as the activation function and 1024, 512 and 256 units in the three hidden layers in the WDNN model, with a learning rate of 0.05.

For a logistic regression problem, the model's prediction is:

$$\hat{p}(Y = 1 \mid x) = \sigma(w^{T}_{wide} [x, \varphi(x)] + w^{T}_{deep} a^{t} + b) \quad (1)$$

where,

- Y is the binary class label
- φ(x) are the cross product transformations of the original features x
- σ(·) is the sigmoid function
- a is the vector of the final activations for the wide model.
- b is the bias term
- w_{wide} is the vector of all wide model weights, and w_{deep} are the deep model weights applied on a^{t}
- $\hat{p}$ is the probabilities vector for the binary class label Y

In the above equation, the binary class label(Y) is the hotel name and x are features of the user, P(Y = 1 | x) gives us the probability match of a hotel given user's features x.

We calculate the logistic loss as follows:

$$\text{Loss} = -y\log(\hat{p}) - (1 - y)\log(1 - \hat{p}) \quad (2)$$

In the above, equation, the loss function focuses on minimizing the error of prediction, where the binary class label(Y) is the hotel name and p is the probability of the class label Y being detected. The calculated loss is back propagated through the wide and deep parts of the WDNN.

Following are the overall training loss graphs for each of the 4 cities:

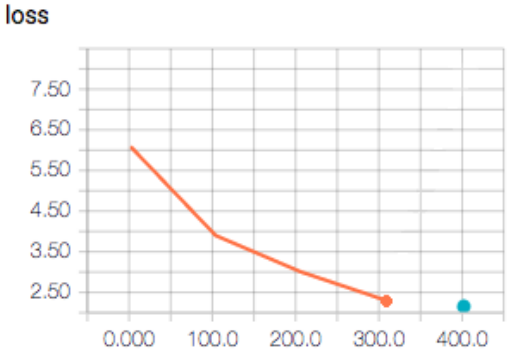

Figure 4: Loss graph for Mumbai

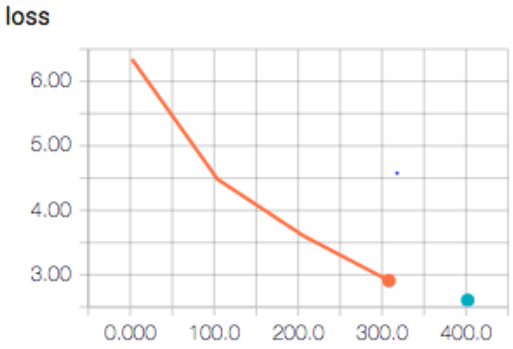

Figure 5: Loss graph for Delhi

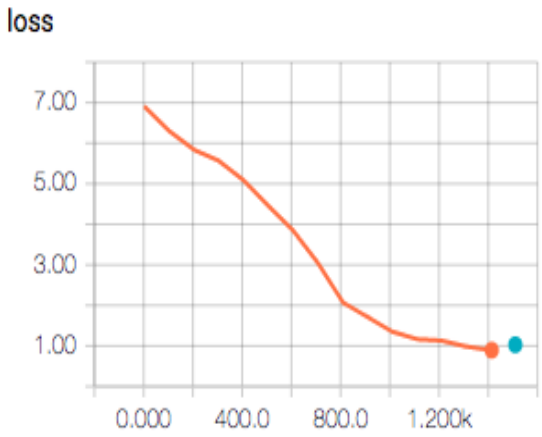

Figure 6: Loss graph for Paris

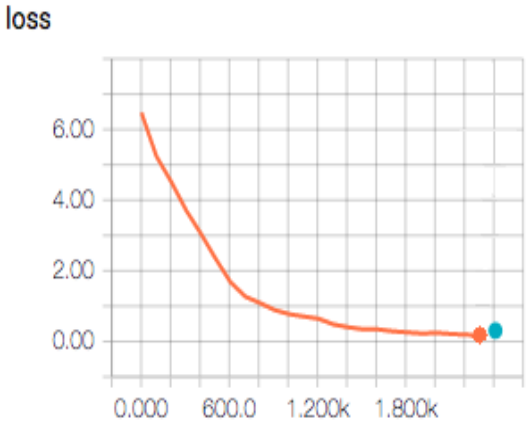

Figure 7: Loss graph for New York

In each of the figures above, the X-axis denotes the number of steps that the model has been trained for and the Y-axis denotes the training loss. The number of steps taken to train the model is directly proportional to the amount of training data (no of reviews) extracted for each city.

We had a relatively smaller dataset for Mumbai and Delhi due to which the loss function converged at 400k steps. For London and New York, the loss function converged at 1300k steps and 2000k steps respectively.

The chatbot was deployed on facebook using IBM's Node-RED to create webhooks and api.ai for entity and intent extraction.

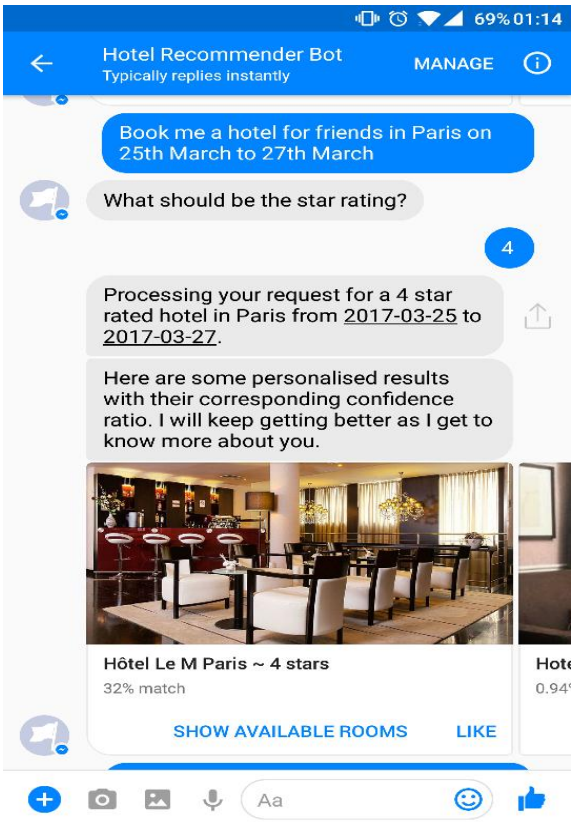

Figure 8: Results from Recommender

The figure above shows an example query for 4 star hotels in the city of Paris, as we can see, hotel results are returned in an in-chat carousel with their corresponding user-match percentage.

V. CONCLUSION

This paper has demonstrated the applicability of Wide and Deep Neural Networks to provide hotel recommendations. We can see from the empirical results that a WDNN can perform effective memorization as well as generalization to provide subjectively accurate recommendations. It can provide hotel recommendations with a percentage match based on the user's profile and previous hotel bookings. In the future, it can be trained to provide complete trip itineraries with the capability to book an entire trip at the click of a button.

REFERENCES

[1] "MyBuys study" available at https://www.digitalcommerce360.com/2009/09/03/product-recommendations-supercharge-online-conversion-rates-stu/ referred on 28/04/17
[2] "Alterra's Marina" http://alterra.ai/ referred on 26/8/16
[3] "Hello Hipmunk" https://www.hipmunk.com/hello referred on 26/8/16
[4] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, Hemal Shah: Google Inc.: Wide and Deep Learning for Recommender Systems

