



TRiPPO

Travel Recommendation System

By

DEEKSHA SINGH THAKUR

JUHI GHOSH

KANIKA KANWAL

(IIIT HYderabad)

INDEX

1. Introduction
 2. Preparing the database
 3. Handling Queries
 4. NLP techniques
 5. Handling Responses
 6. User Interface
 7. Limitations
 8. Future Scope
 9. Bibliography
 10. Summary
 11. Bibliography
 12. Preparing the database
 13. Handling Queries
 14. NLP techniques
-

Introduction

The Chatbot/Virtual Agent is an computer generated, animated, artificial intelligence virtual character (usually with anthropomorphic appearance) that serves as an online customer service representative. It leads a intelligent conversation with users, responds to their questions and performs adequate nonverbal behavior.

TRIPPO is a Travel recommendation System using Chatbot.

Expectation from the Bot : To go beyond statistical text and information-retrieval methods. Contextually meaningful responses will require VAs with articulated knowledge about the possible meanings of words and phrases, connected with each other and with the real-world context.

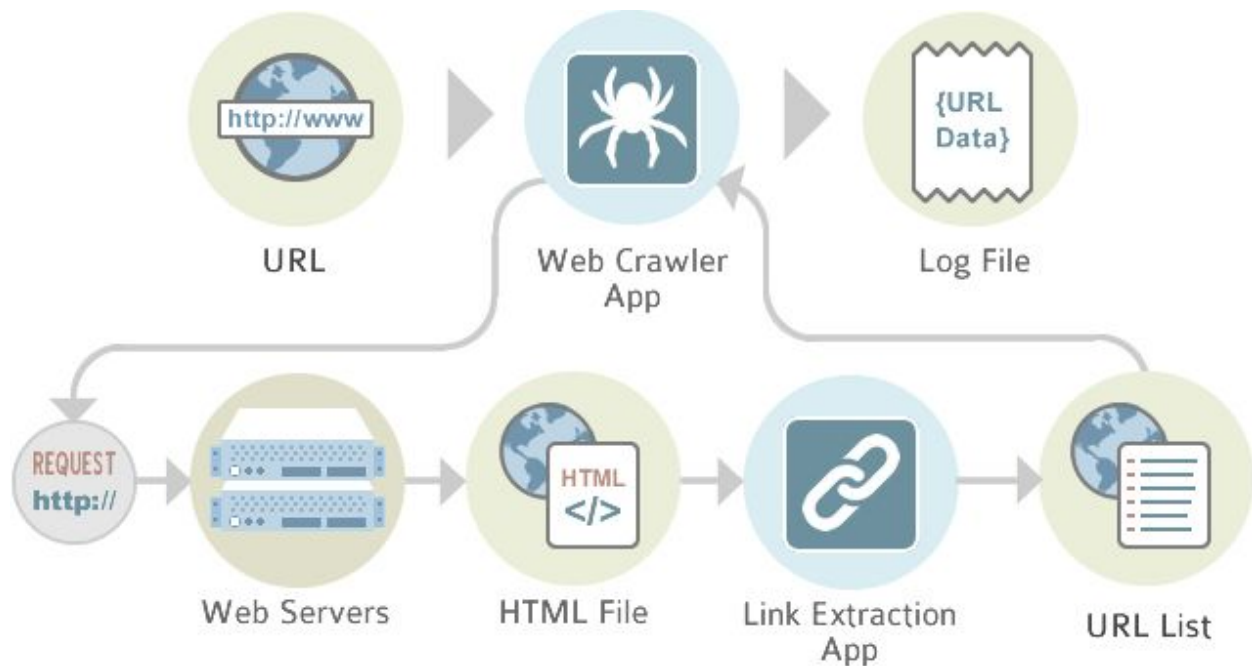
They must continue to improve their language models, but also start to gather information to create richer situational awareness, understand both individual and general context, and become attuned to the intent of the questioner.

Chatbot Design : A simple client - server interaction of user through web interface of Trippo. Query entered by user are sent as request to client, who processes the query in backend, by interacting with the database, and responds to user request with appropriate follow up question or response.

Preparing the database involves three steps:

> **CRAWLING:** In order to get data from a website programmatically, you need a program that can take a URL as an input, read through the underlying code and extract the data into either a spreadsheet, JSON feed or other structured data format you can use. These programs – which can be written in almost any language – are generally referred to as web scrapers. So, crawling is a way of generating a list of URLs from your scraped pages. So, How do crawlers work?

Crawlers are URL discovery tools. You give them a webpage to start from and they will follow all the links they can find on that page. If the links they follow lead them to a page they haven't been to before, they will follow all the links on that page as well. And so on in a loop.



For our project we chose a popular online travel information website “**HolidayIQ.com**”.

The first phase of crawling is done upto 3 levels:

- * the Destination page of the website was crawled in order to get the first set of URLs.

These URLs were then scrapped and all those which belonged to states were stored to be fed for the next loop.

- * for the next iteration of web crawler we fed the stored links of states to the crawler which crawled on the defined links and stored those links which referred to those of cities.

- * Again these links that belonged to cities were fed in to get the sight-seeing places for each city.

> SCRAPING

Data scraping used for extracting data from websites. Web scraping software may access the World Wide Web directly using the Hypertext Transfer Protocol, or through a web browser. Scraping a web page involves fetching it and extracting data from it. The content of a page may be parsed, searched, reformatted, its data copied into a spreadsheet, and so on.

Libraries used for web scraping: We used the lxml library for scraping. lxml is the most feature-rich and easy-to-use library for processing XML and HTML in the Python language.

For our project, we scraped the above discussed links. Since, there were three different kinds of link so we created 3 different types of scraper programs for parsing the states, cities and sight-seeing places.

> DATABASE

Choosing the right database:

We know that the information internet generates is much large than yesterday. Also, that the enterprise capable of extract the value of that data will adopt better strategic decisions and will offer their customers the best products. Therefore, store and analyze the information as quickly as possible will be key. In this area arises **MongoDB**.

MongoDB is the **leading NoSQL database**, let's see why:

- It can store any type of data: structured, semi-structured and polymorphic.
 - Huge performance in data process and scalability.
 - It can process the large amount of information that we generate
 - It supports the current requirements of the applications
 - It empowers enterprises to be more agile growing quickly
 - It is a document oriented database. This means that in only one document is capable of store all the information required for your product. In the fields of the document we can allocate any type of information including arrays and embedded documents. This
-

allows documents to have a very rich and flexible structure.

- We can shape your schema design on-the-fly, leading to much faster development. This is possible because we can do it from the code side, without the need to spending our time in administrative database tasks, as it would be in relational databases.
- Developers have all the same functionalities they have in their RDBMS.

So, considering our above requirements, we chose to go for MongoDB.

The above scrapped information is then stored in a structured format for the project use.

NLP aspect of Chatbot design worked in three phases -

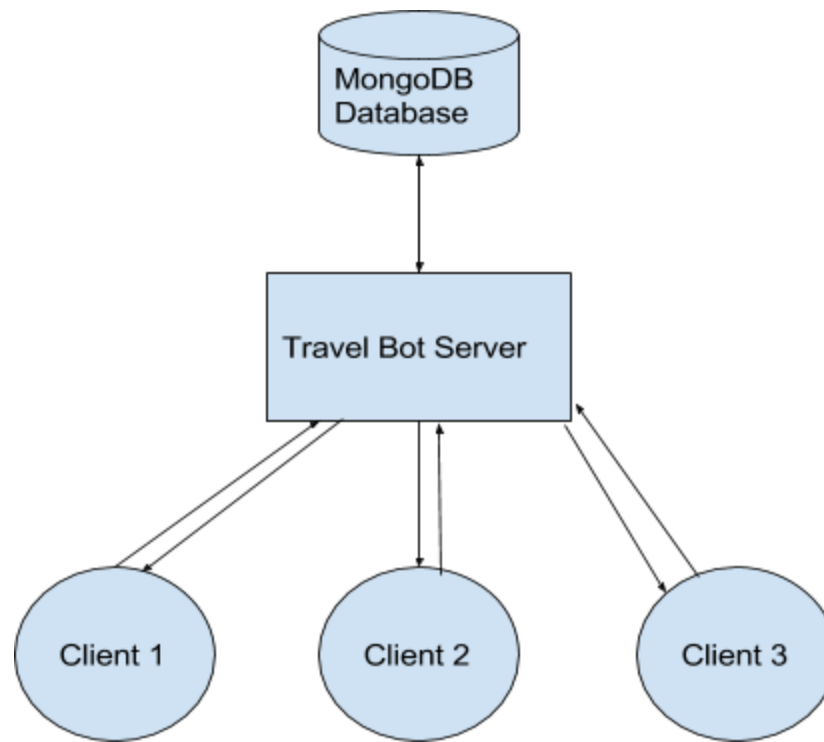
1. Understanding the query : Question understanding defines a range from basic speech-to-text capability through contextual understanding of the meaning of the question.
 2. Interaction - question. Interaction begins with simple one-off question answering, progressing towards more sophisticated modes include phone-tree-like verbal menus, conversational questioning of the user to clarify information, and naturalistic conversation that uses full contextual information.
 3. Response Generation - Responses generated should be relevant to the question, with improvements leading to more fully accurate, useful, and natural responses.
-

Database Structure:

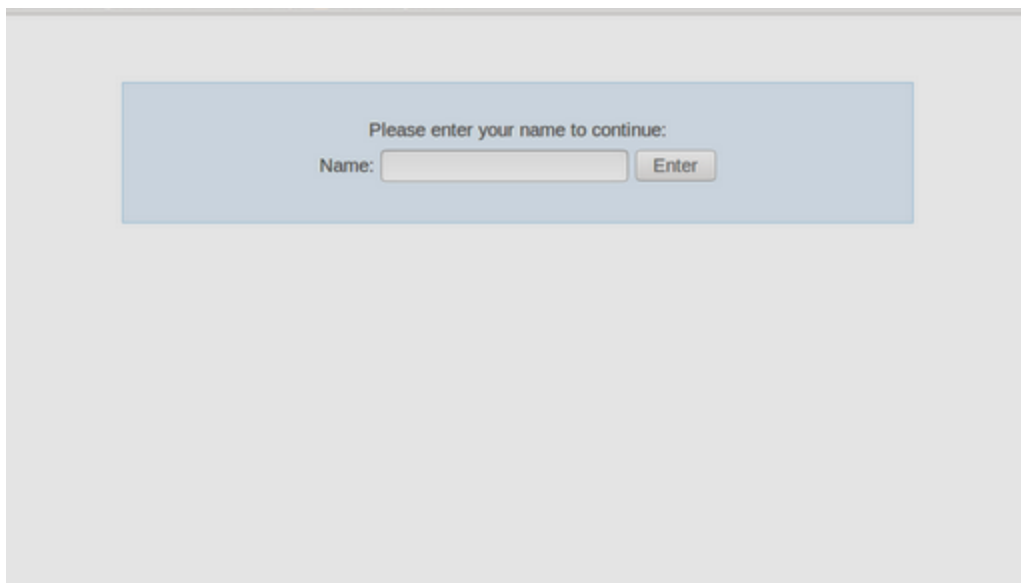
The structure of the database can be given as:

- **Key** : Named Entity
- **Field1** : Intent1/Best_Time
- **Field2** : Intent2/Places
- **Field3** : Intent3/Reviews
- etc.

HIGH-LEVEL DESIGN DIAGRAM FOR TRAVEL BOT

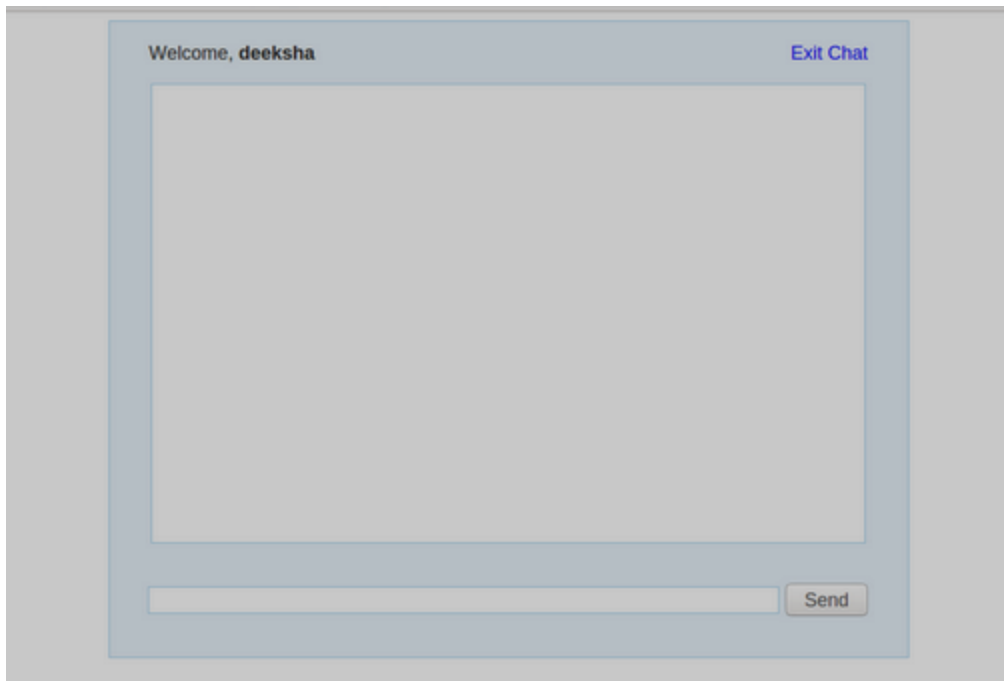


USER INTERFACE



A screenshot of a user interface element. It features a light blue rectangular box centered on a light gray background. Inside the blue box, the text "Please enter your name to continue:" is displayed in a small, dark font. Below this text, the label "Name:" is positioned to the left of a white text input field with a thin gray border. To the right of the input field is a small, light gray button with the word "Enter" in a dark font.

1. Welcome Screen
 2. Chatbot Screen
-



HANDLING USER QUERIES- Handling the queries involve two things:

1. Getting the right intents:

After taking the query we need to parse it such that its semantics are clear to us. By semantics here we mean that we should be able to deduce the intent of the query. Let's take an example for this:

Query: What are the places to visit in rajasthan?

Since, the queries are entered in a natural language so it is easy for a human to deduce what the query demands of. But, doing the same thing by any virtual agent can be challenging. This is the biggest challenge faced by any virtual agent. That is, to figure out what the user actually queries for.

Intent in NLP is the outcome of a behaviour. Intent is important in negotiation to enable a person to open up about the outcome they would like – aside from the behaviour they are displaying to create a desired result.

If we take another example:

Query: Suggest me some places in rajasthan?

Clearly, we can see that both the queries mean the same. But, they use different words (synonyms) to ask the same query. Again, for a human, it is easy to say that both queries are same, but an agent will treat both as different.

Thus, the task of handling queries refers to gathering enough information so that we can give same response to both of our queries.

For the examples we have taken initially, we can see that both queries demand of getting destinations in Rajasthan, so we can conclude that:

Intent: Get destinations in Rajasthan.

2. Getting the right Entities:

Our second task to handling queries involves: getting the right entities.

Similar to the above task of getting intents here also we need to deduce the right entities. Take a look at this example:

Query: What are the places to visit in Rajasthan.

In this query, we know Rajasthan is a place and we need to answer query by stating all the places to visit there. Here, Rajasthan forms an entity and for us humans, we can easily make that out. In NLP, this is nothing but, Named Entity Recognition. That is to locate and classify named entities in text. For our project, these named entities are the names of places, such as Rajasthan in above example.

We take another example such as:

Query: Get me some reviews of this place.

Now here is the problem. In the previous query we could clearly see that our entity is Rajasthan by using any of the NER detection technique. But, just mentioning “this place” means that we need to store another element called the **Contexts**. These contexts are storing the previous entities we used in the queries and thus help in recognising the entities that may not be present explicitly.

So, in total we need to handle these two things: Intents and Entities. We take a look at different NLP techniques to solve the problems we face while handling queries.

NLP TECHNIQUES

1. NAMED ENTITY RECOGNITION USING POS TAGGING -

A Part-Of-Speech Tagger (POS Tagger) is a piece of software that reads text in some language and assigns parts of speech to each word (and other token), such as noun, verb, adjective, etc., although generally computational applications use more fine-grained POS tags like 'noun-plural'.

USE OF POS TAGS IN TRAVEL BOT :

- The incoming queries from user were tokenized into sentences, using NLTK string tokenizer.
 - Each tokenized sentences was sent to NLTK pos tagger, which resulted in POS tagged sentence.
 - Study of POS tags in sentence helped recognize named entities of the query. For Eg. I want to visit Udaipur. Udaipur is NNP tagged word in the sentence.
 - A study of the training queries helped recognize the pattern in user queries, to extract essential information about language structure of queries.
-

-
- Example - Entities “Rajasthan” (a NNP POS tag) usually occurs after prepositions “in”, “of”, “to”. The information helped in Named Entity Recognition

2. NER Named Entity in Travel Bot were required to act as ‘key’ value for our database query. Each named entity is a new record of MongoDB database. We tried and tested several techniques for named entity recognition in a user query.

a. Create static list : List of states, cities and destinations from the pages we have crawled for obtaining data.

b. Studying the pattern in sentences : Usually named entity in a query is preceded by prepositions like, 'in' Hyderabad, 'of' Bangalore, 'to' Chennai, etc. We studied our set of training queries to find the usual pattern among queries to extract named entities in a query sentence.

c. Extracting capitalized words - If the query is written in properly formed semantics named entities can be extracted by extracting the capitalized words of the sentence. (By regex pattern matching - [A-Z].*)

3. HMM

The assumption that the probability of a word depends only on the previous word is called a Markov assumption. Markov models are the class of probabilistic models that assume we can predict the probability of some future unit without looking too far into the past. We can generalize the bigram (which looks one word into the past) to the trigram (which looks two words into the past) and thus to the N-gram (which looks N-1 words into the past)

How do we estimate these bigram or N-gram probabilities? The method used to estimate probabilities is called maximum likelihood estimation or MLE . We get maximum likelihood estimate for the parameters of an N-gram model by getting counts(frequency) from a corpus, and normalize the counts so that they lie between 0 and 1(normalize)

For example, to compute a particular bigram probability of a word y given a previous word x , we'll compute the count of the bigram $C(xy)$ and normalize by the sum of all the bigrams that share the same first word x : $P(w_{n+1}|w_n) = C(w_n w_{n+1}) / C(w_n)$. We can simplify this equation, since the sum of all bigram counts that start with a given word w_n must be equal to the unigram count for that word w_n .

4. SPELL CHECKER

Spell Checker was used in Travel Bot to provide good response to user even if he happens to write wrong spelling in query. Wrongly spelled words may occur due to i) incomplete knowledge of spelling ii) Mistyping words when typing fast/or using keyboard.

We had a dump of all the data we collected from our travel website. This dump was used to form the list of words in my dataset, say *myVocabulary*. Spell Checker worked to find Edit Distance of a wrongly typed word from the words in *myVocabulary*. Our implementation used a spell checker to find nearest word with edit distance 2 from misspelled word.

HANDLING RESPONSES

TRIPPO is an intelligent chatbot that is tamed to respond to user queries in the most desirable user friendly manner. It comprehends the meaning of questions posed to it and maps them to one of the preknown intents that we designed to make planning travel better. As discussed before,

a) We scrapped the data from our parent website HolidayIQ and stored it in a structured format in MongoDB. These data consists of details about states, cities, sightseeing places in cities, best time to visit a city, weather details, mode of travel within and around a city, ratings for a state given by travellers, user reviews about sightseeing places of a city etc.

b) NLP techniques like HMM, N grams, spell checker, POS tagging and keyword extraction are used to understand the meaning of questions asked by user.

c) Next, the task of this section is to analyse the stored data to derive answers to the questions asked by a traveller. We observed and calculated the intents and entities from each user query, these namely are:

1. get description This intent is processed, when the user wants to know the details about a particular place or state. The name of the place serves to be an entity in this query. The response is framed keeping in mind that the user is in search of the geography, historical importance, art, culture, activities, best time to visit, social life of people for a particular destination. His query is whether he may want to visit this place or not. Hence, our chatbot intelligently picks up the highlights of the respective place and portrays in an attractive way to the user.

Example:

User asserts, "Tell me about Indore"

TRIPPO: "Let me give you some details, Indore, the largest city in Madhya Pradesh is situated in India. Some of the attractive places for sightseeing in Indore are Khajrana Ganesh Mandir, Sarafa Bazaar, Shri Omkareshwar Jyotirlinga and Rajwada Indore. The city is brimming with an array of hotels in Indore allowing the travellers to stay and relax during their visits.. Best time to visit is: November December February March "

2. get destinations Processing queries about "Destinations" is the major task of the bot. It is used to address to variety of queries that may be specific or vague . These included,

a) Queries that express the name of a particular location that user wants to visit.

Example:

User asserts, "I want to visit Rajasthan."

TRIPPO: "rajasthan is big state with a variety of cities popular for different activities, these are:

Jaipur, Udaipur, Mount Abu, Jodhpur, Jaisalmer, Ajmer, Ranthambore, Pushkar, Alwar, Bikaner,"

b) User has no preset choices about his taste or liking of places and wants to know about all the places he can see in a city he specifies.

Example:

User asserts, "Best tourist places in Udaipur."

TRIPPO: " Dear Traveller, Come and explore the Popular Sightseeing places in udaipur , these are:

Pichola Lake, Fateh Sagar, Saheliyon Ki Bari, Sajjangarh, Lake Palace Udaipur"

c) User specifies his desire to travel to “**Popular Holiday Destinations**” that can be **beaches , hill or honeymoon destinations** etc in India. He is not sure which is best suited for him and on the basis of our suggestions he will make his mind. Once user makes a choice of a destination the chatbot asks him follow up questions to direct his travel in the desired way.

Example:

User asserts, “Suggest me some beach places in India.”

TRIPPO: “Hey! Please make a choice among these popular beach places :

Panjim, Kochi Cochin, Port-Blair, Havelock-Island, Kovalam, Alappuzha-Alleppey, Neil-Island, Kanyakumari, Ross-Island, Viper-Island, Ratnagiri”

d) The travel bot is also capable of comprehending the context of the previous conversation going on with the user. It responds smartly, if user uses pronouns to refer to a place that he has been talking about in his previous queries.

Example:

User asserts, “ I want to go to Pachmarhi”

TRIPPO: “Let me give you some details,

Pachmarhi is widely celebrated as the sovereign of the Satpura ranges. Located in Madhya Pradesh, it is surrounded by a magnificent hill range; Pachmarhi has traditional memorials for sightseeing and once can also find a lot of othersightseeing in Pachmarhi. How to reach Pachmarhi is also not a problem as there are a number of practical options. There are also a wide variety of resorts in Pachmarhi to help with accommodation.. Best time to visit is: November December February March April.

USER: What are the places to visit there?

TRIPPO: Dear Traveller, Come and explore the Popular Sightseeing places in pachmarhi , these are:

Bee Falls, Dhoopgarh, Pandava Caves, Jata Shankar Cave, Apsara Vihar, Rajendra Giri Sun Set Point, Handi Koh, Dutch Falls,”

3. plan a holiday

This is a special query case. This intent was used to answer the most specific queries from users. For instance, if the user has a plan of visit to a place for a defined period of time, during a particular month or season, the bot is able to follow his specifications and formats a respective reply for the user. The **entities picked up** from this query are, **Place, Duration, Time**.

Example:

User asserts, " Plan my trip to Goa for 3 days in December"

TRIPPO: "You can visit the following places :
calangute, panjim, candolim beach"

4. how to reach

Trippo is capable of guiding users travel by providing suggestions on the mode of conveyance. It scrapes HolidayIQ to extract information on reachability of various famous destinations. Hence when asked about reaching a place by specifying a desirable mode of transport by user, Trippo responds with necessary details of airports, bus stations or railways, as queried. For more information, Trippo provides a travel manual with train schedules etc in a PDF format to the user for reference.

Example:

User asserts, " How to reach Udaipur"

TRIPPO: "You can travel via Bus, Train, Flight or Cab. How would you prefer to go?"

USER: by flight

TRIPPO: By Flight: Udaipur airport or Dabok Airport or Maharana Pratap Airport is the International airport in the city. Direct flights to Udaipur are available from other major cities of the country like Delhi, Jaipur, Kolkata and Mumbai. Some of the flights that are available daily to Udaipur are Jet Airways, Indian Airlines, Kingfisher Airlines and Air Deccan. Udaipur travel cabs and buses are available from the airport to get to any place in the city."

5. get ratings

Several times travellers before making their decision to visit a place want to know the ratings of that place by other travellers who have already visited. This feature makes the traveller comfortable, in knowing the place better and relate to the experiences of other users who rated the place. Trippo collects the ratings of states and cities provided by HolidayIQ travellers.

Example:

User asserts, " Rate Goa on 10"

TRIPPO: " Travellers say,

Average Rating: 5.9/7 Very Good, for Rajasthan

You may also like to see this video review from a traveller,

<https://www.youtube.com/user/TheHolidayIQ>"

6. get reviews

Trippo provides user reviews about a place, when asked for. It uses sentiment analysis to study the polarity of sentences present in first three sentences of the top ten user reviews found on HolidayIQ. Two of the best results showing the maximum positive

polarity are provided to the user as response.

Example:

User asserts, "Give me the reviews of Rajasthan."

TRIPPO: "Dear Traveller Rajasthan Reviews are as follows,

>> There is so much to see, so much to explore, so much to discover and so much to assimilate here. I once read somewhere that if someone wants to see true India then they must come to Rajasthan. This is that one single place which encompasses of so much culture; ethos, tradition, and heritage that I sometimes wonder how could there be an awful lot of varieties in this one locale? Once a person comes here they will go back with a tranche of their heart lying here. And that's the reason people love to visit here again and again. Cities of Rajasthan which should be on your must visit lists are Jaipur, Udaipur, Ajmer, Pushkar, Jodhpur and Jaisalmer. For wildlife lovers Ranthambore national park is a heaven.

>> Very good state and very nice people. Food is spicy but delicious. Must enjoy traditional food and dance also. Very good traditional clothes also purchase over here. Take camel ride in dessert. On the way to Rajasthan many places are there to explore."

LIMITATIONS

a) Data Dependency : Trippo is framing its responses depending on the data scraped from the parent website, HolidayIQ. Any discrepancy in the information provided as response is not verified or corrected. It solely vests on the knowledge that it gathers from the website.

b) Intents can be confusing : Trippo can get confused while responding if the queries posed to it are ambiguous and have more than one intent.

c) Probability of False Positives : There is a chance when Trippo fails to understand the query, and responds with wrong answers. This chatbot is trained on the Markov model using a predefined set of queries that is not huge to support large variations. We need to enrich our training data set of questions subsequently over time, to enhance the degree of accuracy of Trippo's answers.

FUTURE SCOPE

a) We tend to add a feature to our travel bot that is, it **recommends** the user to add more places in his travel plan, that are nearby his current list of **places to visit based on Geo Locations**.

b) Trippo is currently only addressing queries of users on best time to visit a particular place, but we want to make it smart enough to present to the user the weather of a particular place during a defined season of the year. This can be done with **integration of Yahoo Weather API's**.

c) The **database** is defined for the popular destinations in India. It can be **extended to more remote places of beauty** that are yet to be explored by tourists.

d) **Increasing the power of Trippo to understand** and answer to user queries in domains other than the intents we have already explored.

