**1Intelligent Travel Recommender System (ITRS)**

**Savitaa Venkateswaran | Subikshaa Senthilkumar | Sachin Prabhu Thandapani**

**1. Introduction & Motivation**

What do you do when you need a break from your mundane life – You go on a **Vacation**. Everybody loves going on vacations, but we need a considerable amount of time sitting and planning for one. We must go through every hotel, attraction, restaurant, their rating, reviews, attributes and choosing the set of things to do on our budget – The entire process is highly tedious. Planning for a vacation that accommodates all preferences of the traveler without having to look through at least hundreds of websites is close to impossible. Our motivation behind doing this project is to reduce the time spent on planning for the vacation and help travelers spend more time on a vacation that they will love. Our project provides a **Tailor-made Travel Plan** for travelers based on their travel details and preferences. We obtain information from users about the destination, travel dates, budget, amenities they look for in hotels, categories of attractions users would love to visit and their favorite cuisine type. Using this, we generate a plan for the days of travel, **recommending** users where to stay, what to do at what time of the day and where to eat for different meals of the day. This project will be a one-stop tool for travelers planning their vacations, We achieve the same using Data Science tools and techniques which will be explained further in the report.

**2. Problem Definition and Challenges**

For the goals of the project, we have used 3 different recommendation models – 2 Collaborative filtering techniques and 1 Hybrid technique. The motivation behind this was to explore different models in the field of recommender systems and come up with an efficient solution. Restricted Boltzmann Machine (RBM) and Matrix Factorization using Alternating Least Squares technique were the Collaborative Filtering techniques used and Hybrid technique used K-Means for content-based and a K Nearest Neighbours for a Memory-based Collaborative Filtering technique. The major modules of the project include:

**(i)** Dataset collection through web scraping, cleaning and integration **(ii)** Hotel recommendation based on user’s amenity requirements **(iii)** Attraction recommendation for Morning and Evening using categories of tours **(iv)** Restaurant recommendation for different meals of the day using cuisine type **(v)** Integration of all the recommendations and providing a plan for each day of travel.

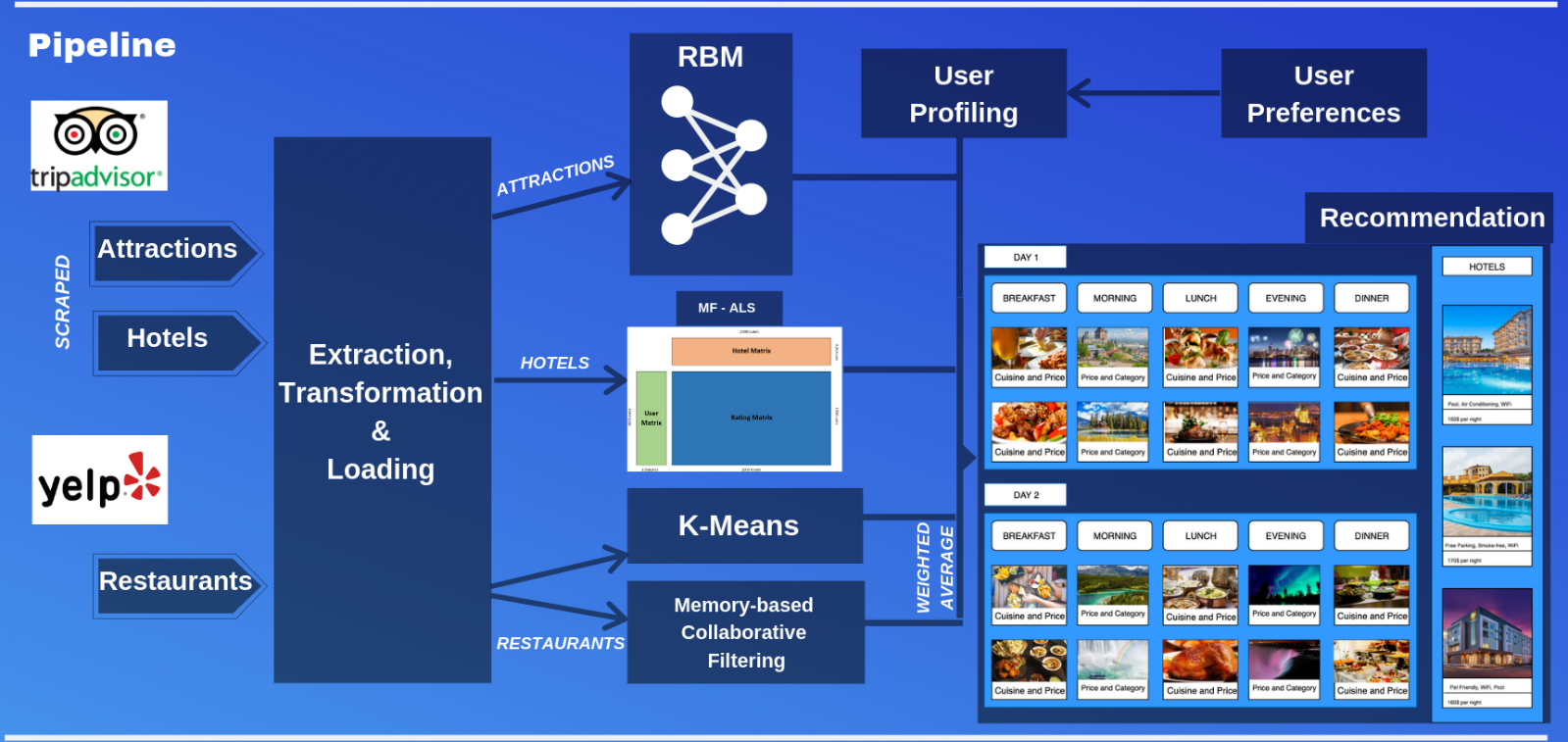
Major **challenges** in implementing the above modules were as follows:

(i) **Data:** The data required to provide the recommendations mentioned above was not available immediately. Web scraping techniques were used to crawl through several attractions, hotels and the reviews on each item and scrape the data from relevant websites. (ii) **Categories** **of Recommendation:** Using different models for providing recommendations for Hotels, Attractions and Restaurants required different methods for data collection, pre-processing and cleaning, different techniques for user profiling and hence thrice the work. (iii) **Evaluating** and improving the performance of the models to obtain relevant recommendations (iv) **Integration** of all the recommendations to get an efficient travel plan was a challenge. (v) **Scalability** of the recommendations provided.

**3. Data Science Pipeline**

**3.1 Data Collection:**

For the project requirements, we needed datasets containing information about details of attractions, hotels and restaurants and their corresponding reviews. There was no dataset immediately available for Hotels and Attractions making Data Collection a very important task for our project. To obtain these datasets, we crawled through thousands of attractions and hotels on **Tripadvisor**. We used ***lxml*** - a python library for processing HTML files using ElementTree API. We had to crawl through hotels and attractions in a different manner due to unique HTML page structures of both categories. Around 3.5k attractions and 35k hotels were collected through scraping tripadvisor. We also created a crawler for scraping the reviews for each attraction and hotel in the dataset and collected reviews close to 33K and ***1M*** for attractions and hotels respectively. Since the websites has a limit on the number of hits per day, we performed the data collection in batches. For the restaurant recommendation, we used data from **YELP dataset challenge**. The Challenge is a chance created by Yelp for students to conduct research and analysis on their data and share their discoveries with yelp. We used the yelp business and yelp reviews dataset from the challenge. The business dataset contained details of all the business on Yelp including restaurants, salons, gas stations, drugstores, etc., without any feature explicitly mentioning the categories of these businesses. The challenge was to extract only the restaurants from the dataset. The techniques used to do the same will be explained later in the report. We got \_\_\_ restaurants and close to \_\_ reviews on the dataset. Fig \_ explains the schema of all the data.

* 1. **Data Collection and Integration:**

Having collected all the data mentioned above, next big task was cleaning the dataset and integrating them. Due to the inconsistencies in the Tripadvisor website, there were missing values for prices, ratings and details about the location for both the scraped datasets.

1. *Hotels*: For the hotel dataset, GeoPy - a python library to geolocate a query to an address or coordinates was used to obtain the latitude, longitude positions of a hotel from it’s address. The missing prices were filled hierarchically with the average price of hotels present in a city, then province and finally the entire dataset based on availability of data. This was based on the assumption that hotels in same locality have similar pricing for competency in the market. Missing ratings were filled out in the same manner. Amenities was the most important feature in hotels dataset and hence, for interpretability, hotel amenities (string) was split, exploded, cleaned and converted into a list of amenities. Continuous unique IDs were provided to users and continuous hotel IDs were provided to hotels.
2. *Attractions*: For attractions, only the priced tours and sightseeing were collected. The locations of the attractions could not be scraped from the website, and hence the address of the attraction was created from the information scraped from websites, and finally Google API was used to obtain the coordinates of an attraction. The missing coordinate values after this step was handled based on the assumption that similar categories of attractions will be present in the same locality. For example, two different water sports will be present next to each other in a city, there will be an area in the city where all the clubs will be present, etc. The location coordinates of an attraction were assumed to be the average location of attraction categories in a city. Missing prices and ratings were also averaged based on city and category of attraction.
3. *Restaurants*:
   1. **Exploratory Data Analysis:**

In order to have a better understanding of the data collected and integrated for recommendations, we performed extensive exploratory data analysis on various features of each dataset. This step was highly significant in helping us understanding the distribution of prices and rating across categories and other attributes in the dataset. EDA helped us arrive at the decision for using specific attributes from each dataset for user profiling by explaining variance in the data with respect to different attributes in the data. The following is the list of few important analysis that led us to the decisions we made:

**(i)** Average rating for different categories of attraction **(ii)** Rating distribution for attraction categories **(iii)** Average Price and rating distribution across different provinces of Canada **(iv)** Price distribution vs Average price for attraction categories **(v)** Province-wise and city-wise number of attractions **(vi)** Distribution of poorly and highly rated restaurants province-wise **(vii)** Rating distribution for restaurants **\*\* INSERT POINTS ABOUT SENTIMENT ANALYSIS \*\***

**3.4 Input and User Profiling:**

To providethe travel plan to the user, we obtain information about user’s travel dates, their budget and their attraction, hotel and restaurant preferences. One of the major challenges in our project was to create a profile for user based on his preferences, A very common problem in providing recommendations to users is the **cold-start problem**. That is, providing recommendation to a new user who does not exist in the database. For a new user, there is no information on what he likes to provide him with recommendations. If we have a website, we can take into account the user’s interaction with the website to provide implicit recommendations, but even then, there is no possibility of providing him recommendations he will like before his interactions. In order to solve this problem, we need to profile the user understanding his preferences. Since, we have 3 categories of recommendations and we used 3 different for the same, we had to use different user profiling techniques:

1. *Hotels*: From EDA, we understood that hotel amenities is the major feature to consider for providing recommendations. We exploded the list containing amenities of a hotel and found top 15 amenities that most hotels have. The user was provided with these amenities to choose from and based on the amenities he chose, hotels were rated on a scale of 5 with rating 5 indicating the hotel containing maximum of user’s required amenities. A 10% of this dataset was randomly sampled and combined with the original dataset containing ratings and reviews on hotels. The combined dataset was then passed to the model for training. This user profiling enabled us to match the new user with the closest user in the existing database who had liked similar hotel preferences.
2. *Attractions*: Travelers have unique preferences in visiting different categories of attractions. For example, a travel group might like to visit beaches, water and adventure sports. Another group of travelers might like to visit museums, parks and sightseeing. This observation was leveraged to perform user profiling for Attraction recommendations. 20 Most frequent categories of tours were taken and provided as options to users. User was asked to select and rate at least five of the provided categories. This information then used to match the new user with a user in existing database sharing similar preferences, in essence, rated attracted categories in similar fashion. The recommendation obtained for the matched user in the database was provided to the new user as his attraction recommendations.
3. *Restaurants*:

After User profiling, the models are trained and evaluated to obtain best performing models. Then the recommendations are filtered and integrated to an **Interactive Python Notebook**.

**4. Model Training Methodology and Evaluation:**

As mentioned earlier, 3 different models were used to obtain the recommendations. In this section, we will discuss about the models used in detail, the input for the model, training and evaluation of the model.

1. **Matrix Factorization using Alternating least Squares technique:**

Matrix factorization is one of the most used Collaborative Filtering technique after Netflix Prize Challenge. Matrix Factorization uses the user-item interaction matrix to learn the embedding among the interaction to predict the ratings that will be given by users to unseen/ unused items. Alternating Least Squares (ALS), a matrix factorization algorithm, runs in a parallel fashion. It runs the gradient descent of multiple partitions of training data in parallel. It is also highly scalable and thereby used for large collaborative filtering tasks. It is simple and deals with solving the sparseness issue of the ratings data. Also, ALS can be used in distributed file systems.

*INPUT*: The hotel data containing information about user, hotel and user’s rating for the hotel was used as input for training the MF-ALS model. The dataset also contains 10% sample of the new user data created using his preferences of hotel amenities. The remaining 90% sample used to obtain recommendations for new user.

*PARAMETERS*: The ALS model was trained with three different ranks (i.e.) 4, 8, 12. Ranks are also known as the latent factors. Latent factors are the features projected from user-item interaction in the lower dimension latent space. To represent user preferences in lower dimension space, latent factors are used. The ALS model was trained with three different ranks, maximum number of iterations as 5, regularization parameter as 0.01 and three attributes namely user id, hotel id and the rating given by user for the hotel.

*EVALUATION*: Root Mean Square Error (RMSE) was calculated for each case with different ranks and the model with lowest RMSE was used for recommendations. RMSE is calculated by finding the difference between predicted and existing rating, squaring them and then finding the square root of that value. The model with least RMSE denotes that the user-item interaction embedding has been learnt properly and the model is able to predict user ratings to items more accurately.

*RECOMMENDATION*: The saved model was used for predictions. Based on user preference of amenities, ratings were provided to hotels. 10% of the sample data containing user preferences was combined with the actual training data. Remaining 90% of the sample data was used for predictions. The data used for predictions contains the hotels not visited by the user. The trained ALS model is used to predict the ratings that the user will give for the hotels they haven’t visited.

1. **Restricted Boltzmann Machine:**

Boltzmann machines are stochastic and generative neural networks capable of learning internal representations and are able to represent and solve difficult combinatoric problems. RBM’s are non-deterministic Deep Learning models with only two types of nodes - **visible** and **hidden** nodes. A regular Boltzmann machine has connections between all hidden nodes and visible nodes and amongst themselves. **Restricted Boltzmann Machine** is a special class of BMs and is restricted in terms of the connection within the same category of nodes and if fully bipartite. There are two other layers of bias units (hidden bias and visible bias) in an RBM. The hidden bias of RBM produces the activation on the forward pass and the visible bias helps RBM to reconstruct the input during a backward pass. The reconstructed input is always different from the actual input as there are no connections among the visible units and therefore, no way of transferring information among themselves. The model is trained to achieve the input after the reconstruction phase.

The inputs are multiplied by weights and hidden bias is added to them. This activation is then passed into a **sigmoid** activation function. This is the **forward-pass**. During the **backward-pass** or **reconstruction phase**, the activation function outputs act as the new input, multiplied by weights, added to the visible bias and then passed through thesigmoid activation function. The output of the reconstruction phase is then compared with the input to the forward-pass to calculate reconstruction error. The model is trained by reducing the reconstruction error using a gradient-based **Contrastive Divergence** algorithm. Contrastive Divergence is the difference between two Kullback-Liebler divergences of the input and reconstructed output. The algorithm tries to bridge the gap between the input and the reconstructed output while updating the weights to do so.

*INPUT*:The attractions dataset was used to create the user-item interaction matrix, also known as **rating matrix.** The rating matrix contains *n* rows for users and *m* columns for each attraction present in the dataset. The ratings provided by the user *i* for attraction *j* was filled at (*i,j*) position of the matrix. The positions without any user-item interaction was filled using a 0 rating. To avoid overfitting, the dataset was split for training and validation and each row of the matrix was then passed to the visible layer of the RBM model for training with weights, hidden bias and visible bias initialized randomly.

*PARAMETERS*: The model was trained for different combinations of parameters including number of epochs, learning rates, batch size, number of rows for training and number of hidden units. Trained best model for every combination tried was store for future usage.

*EVALUATION*: The model performance was evaluated using the Contrastive Divergence algorithm and the model with the best performance, i.e., least divergence between input and reconstructed output at visible layer was saved. The model with lesser divergence from output suggests that RBM has learnt well the embeddings between user and the attraction interactions so that it can replicate the input row.

*RECOMMENDATION*: The saved model was used to obtain recommendations for all the users in the existing dataset. Then the traveler is matched with the closest user in the dataset based on attraction category preferences and the recommendations are provided obtained for the traveler.

1. **K-Means + Memory based Collaborative Filtering Hybrid technique:**

**5. Output and Data Product**

The data product for the project was an Interactive IPython Notebook created using **ipywidgets** – an Ipython library for creating widgets in a notebook. Inputs from traveler containing travel details, budget, their hotel, attraction and restaurant preferences were obtained through interactive widgets as shown in figures \_, \_, \_. The final recommendation for each category of recommendation was handled differently and finally integrated into one single **Data Product**.

1. *Hotels*: After obtaining recommendations from the Matrix Factorization model, the result was filtered based on budget and location. The price was multiplied for the number of days of stay and result was sorted and filtered. Details of 5 hotels with top recommendation scores were provided to the traveler as Hotel recommendations to choose from. The result was consistent with the user’s amenity requirements. The final output for hotel recommendation is shown in figure \_.
2. *Attractions*: The result from RBM was filtered based on location and budget. Challenges in providing this recommendation was taking into account distance between attractions and the time of day to visit the attraction. Few attractions and tours will be better for morning visits and few tours like night trekking or watching a sunset would be better to go for in the evening. To solve this problem, NLTK’s Wordnet was used to find the synonyms of evening and night, and these keywords were checked for in the name of the attraction. This enabled us to distinguish **Morning** and **Evening** attractions. Secondly, a person going to an attraction in North Vancouver would not want to travel all the way to Surrey to visit the second attraction. Hence, distance between consecutive visits for attractions had to be considered. The top recommendation with highest score was chosen, distance was calculated between every attraction to our selection and the closest attraction with next highest recommendation score was chosen for next recommendation. The list of recommendations provided were kept track of so that the recommendations were not repeated. This process repeats for all the days for which the traveler will be staying at the destination and the final output will be provided as shown in figure \_.
3. *Restaurants*:

**Major Challenges in building the Data product:**

**(i)** The images of hotels and restaurants provided in the final output could not be scraped from the website. Hence, **google\_images\_download –** a python library for downloading google images was used to obtain images for recommendations based on their name. **(ii)** Using ipywidgets for interactive output in jupyter notebook and to parse HTML commands to provide final output. **(iii)** Efficiently integrating all three recommendations to provide a single final **Data Product.**

**6. Learnings**

1. Gained experience in learning and implementation of 3 different recommendation algorithms – RBM, Matrix Factorization using ALS and a hybrid recommendation algorithm using K-means and memory-based Collaborative Filtering.
2. Understanding data requirements of the project and collecting the data required to create the recommender system.
3. Importance of Exploratory Data Analysis for understanding the data was realised and carefully performed.
4. Explored options to create a User Interface using Jupyter Notebook. Explored ipywidgets, an IPython library for creating interactive widgets in Jupyter Notebook.
5. How to communicate our findings and project to someone through verbal, written and video communication.

**7. Conclusion and Future Work**

The project’s aim was to provide tailor-made travel plans for users based on their travel details and preferences. The aim of the project was successfully accomplished by exploiting various Data Science tools and techniques and the recommendations were provided to the user. The scope of the project is really wide and hence there is a lot of room for improvement and future work. Few suggestions are as follows:

1. The data currently collected for the project was restricted to Canadian hotels, attractions and restaurants and recommendations are provided for only Canadian vacations. This can be scaled for multiple travel destinations around the world and recommendations can be provided for various parts of the world.
2. The data was scraped from tripadvisor website. There are several other websites that can contain more relevant information on the items and hence can be used.
3. The project can be extended by creating an interactive web application to provide recommendations instead of using interactive python notebook to get information from user. Using the web application, interactions of users can be collected to improve the user experience and provide implicit recommendations.
4. The recommendations provided can also be improved by taking into account the number (solo or a group), type (professionals, friends, family with or without kids). For example, having kids during the travel, we can eliminate under 18 recommendations and emphasize on suggesting kids-friendly places like theme parks.
5. In our project, three recommendation algorithms were used. Various other recommendation algorithms can be used and fine-tuned to improve the accuracy of the recommendations provided. For instance, Deep Belief Networks or Auto-encoders can be used instead of RBM to improve the recommendations. There are carious ensemble recommendation algorithms that can be exploited to provide highly accurate recommendations.

**Project Summary**

|  |  |
| --- | --- |
| **CATEGORY** | **POINTS** |
| **Getting the data –** Dataset downloaded from YELP dataset challenge | 1 |
| **ETL –** Used spark to clean and perform ETL steps on dataset | 3 |
| **Problem –** Problem statement was focused on helping both business owners and users get useful insights and recommendations | 4 |
| **Algorithmic work –** Performed NLP operations on reviews and used spark’s MLlib to train different models and get required results | 3 |
| **Bigness/parallelization –** Total dataset size of 7 GB with 6M reviews, 1.5M users and 200K business was used | 4 |
| **UI** | 0 |
| **Visualization –** Created visualization using python libraries and Tableau | 3 |
| **Technologies –** Python, Spark, MLlib, Tableau, Plotly, | 2 |
| **Total** | **20** |