Matt Usifer 4/28/2016

Artificial Intelligence – Project Report

Fast Blue Train: Finding the Optimal Route

Introduction

Travel planning is a vital and innate human skill, and this has been the case throughout all of human history. Though humans have a well-honed ability to plan a route between locations, that ability has been challenged in recent years by the multitude of options for transportation introduced in the modern era: public transit, personal automobiles, personal bicycles, taxis, taxi-like technology start-ups such as Uber, and the list continues to expand. When making the choice of which route to take between two places in the modern age, the goal has not changed despite the increase in optionality. One still seeks to choose the optimal route – the route that is fastest, cheapest, and meets other high-level criteria such as safety and convenience – but the increasing amount of options to consider when making this choice has also increased its difficulty. Some of these modern modes of transportation cost more than others, some take more time, and the cost and availability of these modes is subject to change in unpredictable ways based on external factors such as the time of day and the state of the organization offering the service.

With the advent of the Internet and GPS technology, travel planning has become a much simpler process. Using online navigation services such as Google Maps, users are empowered to find the fastest route between two locations using the 4 most popular major modes of transportation: Driving, Walking, Biking, and Public Transit. However, there are generalizations made in this process that prevent these services from addressing some of the most basic realities of travel in a modern American city. For example, these services always assume that the user is negligibly close to the location of their car or bicycle, which is almost never the case. Additionally, they do not take into account the user's budget and the cost of the route.

Fortunately, many relevant services have exposed themselves via Web APIs. These interfaces allow programmers to incorporate data and functionality from these 3rd party technologies within custom applications, and thereby create a unique opportunity to extend the functionality of these services to fill in the gaps noted above. This work seeks to achieve these goals through the creation of a route recommendation system called *Fast*

Blue Train, and it also seeks to identify opportunities for further advancements that can be made in this area.

Relevant Work

A considerable amount of recommendation systems have been created to help users choose among a large set of options. This type of work has been applied to a wide variety of disciplines, such as choosing a TV show [Kurapati et al], choosing a product [Wang], and also to route recommendation (CrowdPlanner [Su] and R3 [Wang et al]). CrowdPlanner cleverly addresses the route recommendation problem through crowd sourcing - gathering popular routes and suggesting those routes to the user. This recommendation system is centered on the assumption that drivers who live in a given area will choose routes in that area that are ostensibly better than the routes suggested to a traveller via Google Maps or a similar service. The main idea is that some routes, though they look to be ideal on a map, are riddled with obstacles that only a seasoned local driver would know to avoid such as traffic lights or traffic congestion. This is a logical idea, but does not examine any modes of transportation outside of driving in a personal car and it does not address any user preferences outside of minimizing the travel time to destination. R3, on the other hand, aims to provide a recommendation based on real-time traffic information, hoping to overcome the limitations of those route recommendation systems that do not account for unexpected traffic. This is another leap forward in route recommendation, but it is also limited to automobile transportation and it does not account for a user's monetary budget.

In the following section, I present the architecture of the *Fast Blue Train* route recommendation system that I created in order to address the issues outlined above. The section thereafter will outline some of the higher-level findings that I gleaned from this project, and I conclude with some suggestions for future work in this area.

Fast Blue Train: Architecture

I. Technology

From a technology standpoint, Fast Blue Train is a single page web application designed with a focus on efficient computation and rendering. It relies on Google's AngularJS application framework to organize its different software components, arrange their level of interaction, and to enable a responsive user interface in the browser. The server-side logic is written in Clojure, and the client-side logic is written entirely in ClojureScript, a functional Clojure-like lisp that compiles to JavaScript. As implied above, the web application exists in only a single page, and all user-input and application output is handled therein. It is from this technology that Fast Blue Train is given the ability to accept and record user preferences, aggregate route information, and organize routes by given heuristics. The entire architecture is outlined in Figure 1.

This application exists in the form of a *web application* as opposed to a *desktop application* in order to make it convenient and accessible to the general public.

II. User experience

The user experience within this web application can be broken down into 2 parts: the initial state and the post-initial state. The initial state consists only of two search boxes, labeled 'Start' and 'End', as well as two buttons, labeled 'Get Directions' and 'Settings'. The user can click the 'Settings' button to set the following preferences:

- Bike Location
- Car Location
- Car MPG
- Budget for trip
- Has Transpass

The application uses the Google Maps API to enable auto-complete in the start, end, bike location, and car location input boxes.

When the user clicks 'Get Directions' to begin the calculation phase of the application cycle, he or she will also trigger the post-initial state, which consists of all of the

elements in the 'initial-state', along with a map and a results display pane. The results of the calculation phase are rendered both in the map and organized in the results display pane.

III. Optimal Route Calculation

The application begins its calculation phase by assembling all responses that will be sent out to Google Maps and Uber. The maximum amount of requests are sent if the user has filled out all of 'start', 'end', 'bike location', and 'car location'. In this case the following requests[1] will be sent to the Google Maps API[2]:

- Origin -> Destination via Walking, Transit, and Driving[3]
- Origin -> Bike via Walking, Transit, and Driving
- Origin -> Car via Walking, Transit, and Driving
- Bike -> Destination via Bicycling
- Car -> Destination via Driving
- Bike -> Car via Bicycling
- Car -> Bike via Driving

Additionally, the following requests will be sent to the Uber API:

- Time-to-pickup from origin
- Estimated cost between origin and destination
- Estimated cost between origin and bike
- Estimated cost between origin and car

Once the responses from all of these requests have been received by the application, the route optimization calculation begins. The application will aggregate key statistics from

[1] 'Start' and 'end' are both required inputs, but if the user fails to fill out 'bike location' or 'car location', the associated requests will be eliminated from the outgoing batch of requests.

[2] Since the Google Maps API will not allow more than 10 requests per second, the application will inject a delay between requests if it hits the rate limit

[3] Driving is requested from the Google Maps API as part of the initial leg of a route to account for Uber estimates

the legs of high-level route – namely *cost* and *time*. It uses the cost data gained from Uber to incorporate costs from corresponding legs of a given route, and it uses the user's car MPG plus the average gas prices in Philadelphia[4] to incorporate cost from any 'Driving' leg of a route. Furthermore, it will add $1.80 for any 'Transit' leg of a route if the user does not have a SEPTA trans-pass to account for the cost of public transit in Philadelphia. With regard to time, the application will aggregate all estimated times learned from the Google Maps API for all legs of routes. The full duration of an Uber route is estimated by adding the 'Driving' estimation for the route given by Google Maps to the time-to-pickup estimation for that route given by the Uber API.

When all costs and times have been calculated, the arrangement of the routes becomes very straightforward. Routes that exceed the user's budget are removed from consideration, and the resulting collection of routes is ordered by time. Any two routes with an identical estimated duration are secondarily ordered based on their cost.

Findings

Artificial Intelligence on the Internet

This project was implemented with an intention of building an application that would seamlessly combine relevant data from different regions in the Internet and use that data to make recommendations to a user. For route planning, once all of the data is present and accessible in memory, making a recommendation is relatively trivial. The application can apply the user preferences to sort all available routes in the optimal order, thereby providing the necessary services to the user. By far the most challenging part about building this application was obtaining, aggregating, and combining all relevant data. Data points that come out of the Google and Uber API are vastly different in structure and content from one another, and learning how to parse and interpret the desired data from these structures is often impossible without referencing some form of human readable documentation available about the API. Furthermore, building valid requests that can be these APIs is also an unpredictable endeavor. The authentication methods are vastly different, and the desired endpoints are not easy to find without a reference to

[4] The application is designed only for routes within Philadelphia due to the lack of readily-accessible data pertaining to both gas prices and the cost of public transit in different areas

documentation and an ability to interpret the responses that are returned from the server. It is extremely common that a malformed request will trigger a generic '404' error from the server with a more helpful message provided in plain English in the body of the response. If an artificially intelligent machine is going to interact with the huge amounts of data readily available on the Internet, it will need to be able to learn how to conduct those interactions with Web APIs. As human programmers interacting with this data we are able to access documentation and interpret the natural language responses to guide our behavior, but a machine that is aware of how to process language, where to look for documentation, and how to parse data out of the response does not exist.

To make matters more complex, these APIs are constantly changing. The structure of responses, endpoints, or valid requests is subject to change at any given time with only a warning provided in natural language by the vendor either on their website or via email. If an artificially intelligent machine is expected to maintain an API interaction for a long period of time it will have to learn how to adapt to these changes by reading the messages provided by vendors and modifying it's interaction accordingly.

Future Work

There are several elements that I plan to add to this application in the coming months in order to improve its usability, efficiency, and extendibility.

I. More User Information

The application could be recording several additional attributes about the user. A few of these potential attributes are Age, Weight, and a preference *against* or *for* certain travel modes. Age would be useful in the case of a senior citizen, for example, whose preference profile one might want to augment with a preference against bicycling and/or long walks. Knowledge of a user's weight would provide a method of estimating how many calories would be burned on each route. Using this information, we could then add an additional preference for (or against) routes that burn more calories. Finally, a preference for or against certain modes of transportation is common among travelers. Some people simply do not trust Uber or do not like taking public transportation, and these sorts of preferences could be factored into the optimal route calculation relatively easily.

II. Efficiency Improvements

The second category of elements that would improve the quality of the application is centered on performance and efficiency improvement. To start, results from API calls should be cached to avoid sending duplicate requests to the APIs. Additionally, the batch of outgoing API calls could be pruned down to only those necessary based on user preferences. For example, if the user has no budget and no trans pass, it doesn't make sense to send API calls to Google Maps for public transit.

Finally, there is an opportunity within the application to learn about repeat visitors to the site. If the application were modified to allow users to sign in and save their preferences, it could also learn what types of routes a given user generally chooses. Given enough of this data, the route recommendation system could make better recommendations that are catered to the user based on their historic route selections.

III. Expansion

The application could be further expanded in a variety of ways. Lyft, another popular taxi start-up that is similar to Uber, also has an API that could be incorporated into the application. The scope of this application should also be expanded to outside Philadelphia. This may mean first expanding into specific cities before creating an application that is truly flexible enough to calculate routes between any origin and destination. This would require that gas prices and the price of public transportation are available programmatically (i.e. via an API), which is currently not the case. The final expansion that will eventually be added to this application is an expansion to consider air travel and rental cars. These are two extremely common forms of transportation that could easily be incorporated into an application that had a wider scope by design.

Conclusion

This project was an exercise in applying cutting-edge technology to an age-old problem: planning an optimal route. In the form of a web application, this technology gains access to a plethora of relevant data and can be made readily available to all who are interested in using it, but it is not clear how it can be expanded to a true artificial intelligence. An artificially intelligent web application would need to overcome several obstacles in order to dwell on the Internet without human intervention. To name a few of these obstacles that are yet to be fully overcome, this machine would need to process natural language from free text, it would need an idea of where data is located, and it would need a clear goal of what service it was providing. Until those and several other lower-level needs are met, web applications will be dependent on human intervention in order to function properly.

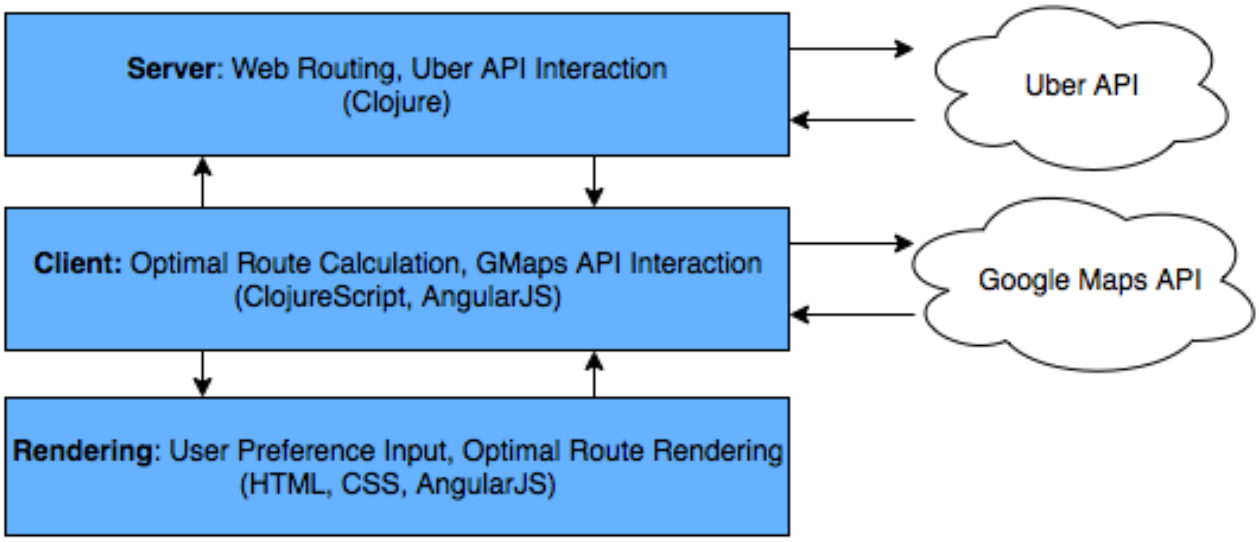

Figure 1. Architecture of the Fast Blue Train route recommendation system

References

Kurapati, Kaushal, Srinivas Gutta, David Schaffer, Jacquelyn Martino, and John Zimmerman. *A Multi-Agent TV Recommender*. N.p., 2001. Web.

Su, Han. *CrowdPlanner: A Crowd-Based Route Recommendation System*. University of Queensland, Australia, 2013. Web.

Wang, Henan, Guoliang Li, Hu, Chen, Shen, and Wu. *R3: A Real-Time Route Recommendation System*. N.p., 2014. Web.

Wang, Pei. *Recommendation Based on Personal Preference*. N.p., 2003. Web