Tripster

An web application to make your road trip planning easy and enjoyable

Minfa Wang[*]
California Institute of Technology
mwang5@caltech.edu

Shenghan Yao
California Institute of Technology
syao@caltech.edu

Junlin Zhang
California Institute of Technology
jzhang5@caltech.edu

ABSTRACT
Tripster is a web application that provides an aesthetical and intuitive interface to make road trip planning easy. Tripster is developed in JavaScript, HTML and Python with the help of Google Maps API JavaScript, Google Maps API Python, Yelp API and JQuery&AJAX.

Categories and Subject Descriptors
D.2.2 [**Design Tools and Techniques**]: User Interfaces; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*; F.1.2 [**Modes of Computation**]: Parallelism and concurrency; H.3.3 [**Information Search and Retrieval**]: ClusteringInformation filtering-Query formulation; H.3.5 [**Online Information Services**]: Web-based services

General Terms
Clustering, Navigation, Recommendation

Keywords
K-means, GAE, Google Map API, Yelp API

1. INTRODUCTION
Road trips with friends and families are always one of the best moments of the life. Nowadays people will either spend a lot of time researching before the trip or donâĂŹt plan ahead at all. For the people who would like to plan ahead, choosing the best route and attractions along with your origin and destination takes a lot of effort. Sure you can ask for friends' advice if you happen to have friends who have been to all the places that you are going to, or if you want, you can Google tons of times to find other people's advice, or if you really have time, you can study every single city along the route with Yelp and Google. But there must be a better way to do it. For example, you are planning to road trip from Los Angeles to New York City, and you want to find hotels to stay for your entire family during the prime summer time. In this scenario, you might have to find a nice place to stay in a less populated area. How would you find it? You can surely Google or yelp 100 times to find nice places along the routes during that area but that is really a waste of time. Another problem with current road trip planning is that people only know the most popular attractions and there are a lot of local attractions and that is probably the best way to experience the new city. For example, everyone knows that they should stop by Golden State Bridge to take a selfie, but there might be a wine festival going on during your stay at San Francisco, and if you are a wine lover, there is no reason to miss such opportunity.

[*]Graduate Student at Caltech

One of my personal examples, a couple of years ago when I was planning our road trip from Iowa to Rocky Mountain National Park, first I will first need to look up Google map to find a few possible routes. Next I need to plan how many days to spend on road. Then according to my preference, I manually adjusted the route. The real tricky part is to plan overnight stay. If I want to stay somewhere overnight I will need to first figure out where exactly to stay and then I will look up hotel information on Yelp. But the tricky part comes in here, due to unfamiliarity with local amenity, the city I happen to choose to stay doesnâĂŹt have a lot of good hotels but the town 10 mins away from it has lots of great restaurants and hotels. But there is no way I can know this information without doing lots of researches on cities and peopleâĂŹs travel blog.

To solve the above issues with road trip planning, we came out with Tripster which is a powerful web interface to make road trip planning easy and enjoyable. It powers with a customized Google Maps interface. Users enter the origin and destination and users also have the chance to tell Tripsters about the attractions users are interested (i.e. local festivals, night clubs, Chinese Food, etc) Based on above information, Tripster will help figure out the best attraction candidates to match your preferences along the possible routes. In order to not overwhelm users with too much information, the information will be stored inside clusters along the routes. Users can see attractions in each cluster by clicking and zooming in. Users are able to view each single attraction's information including address, phone number, Yelp review score and review number. If more research is desired, they can open the link to Yelp directly. Then if the store still holds up well after all these research, users can add the store to routing, After selecting all of the attractions, Tripster is going to route the best route based on attractions that users choose.

2. RELATED WORKS

2.1 TripIt

This application helps travelers to manage schedule. The major product is a mobile app. It seems the targeting customers for this company will be the business people, who are not going to somewhere to enjoy a break, but rather have some rigorous plans to execute. It looks like it has many functionalities to help manage time punctually and accurately. However, the design is kind of too complex to get a direction of where heading to. Also, purely a time management tool is not interesting enough to attract users (or at least road trippers).

2.2 TripAdvisor

This website is famous among travelers. In this website, people can search for attractions, restaurants, etc. They can book hotels, write reviews and a lot of other stuff. While it is definitely very useful for travelers, the focus of that website is certainly different from ours. When the user arrives at the city of destination, and wants to explore deeply into a city, then this website could be perfect to that person. However, what should the person do and see in the middle way of driving? Searching on TripAdvisor for nearby points will be less useful. That's where our application comes into play. We

2.3 RoadTrippers

This website is dedicated to providing trip planning services to roadtrippers, and is the one whose service is most similar to ours. In fact, they have done a pretty good job in user interface design. The way this it works is that user first chooses the trip origin and destination, then there will be a route shown, with lots of dots scattered around, representing interesting points. Then users can further choose the points in the map and customize the trip. Its basic functionalities are almost complete. However, we think that although this website could be very useful for the people who already know well about a place, and already know where they will stop by in the middle of the trip, this website could be overwhelming for people who have little prior knowledge about a place. And we think for readtrippers, the latter case is not just the minority. We are trying to move one step further, and give recommendations of where they could stop by just based on some general preferences that they choose.

3. BASIC IDEA

The basic idea of our implementation works as follow.

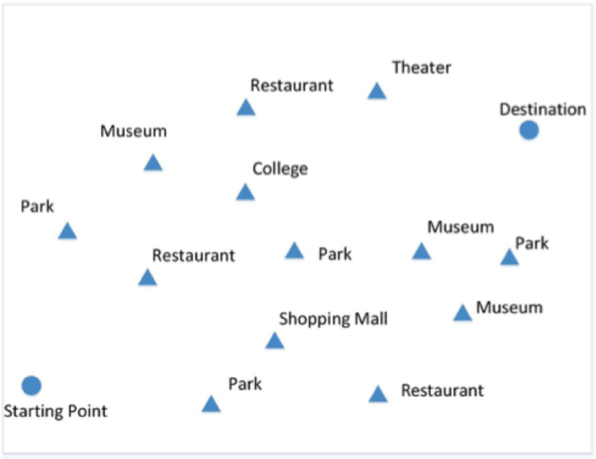


Figure 1: simple map with points scattered around

Initially, in a user's (road tripper) mind, with not much knowledge about the middle way, will have an origin and destination point. Then in fact, there will be potentially interesting points like museums, parks and restaurants in the middle way.

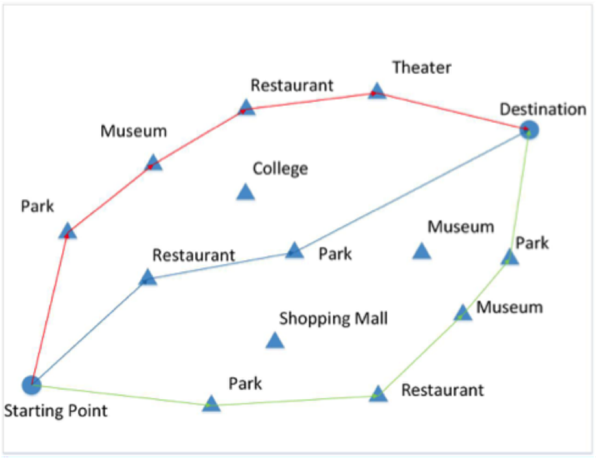


Figure 2: three potential routes

With this information, we could first blindly search for path to go from origin to the destination, and in the backend, list all the potential routes. There will be certain criterion to choose the route, like time spent and total distance. We will not be so rigorous here in setting those restrictions.

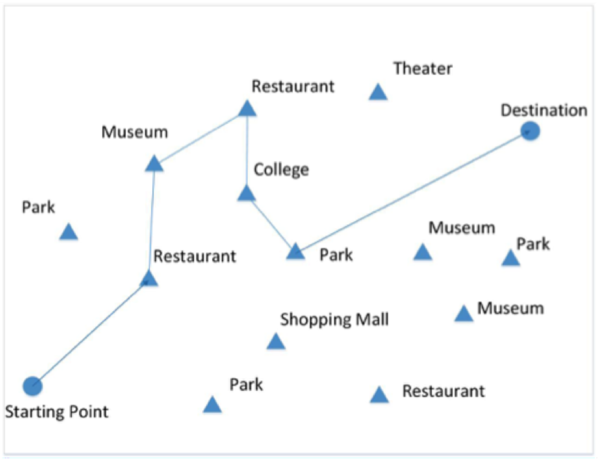


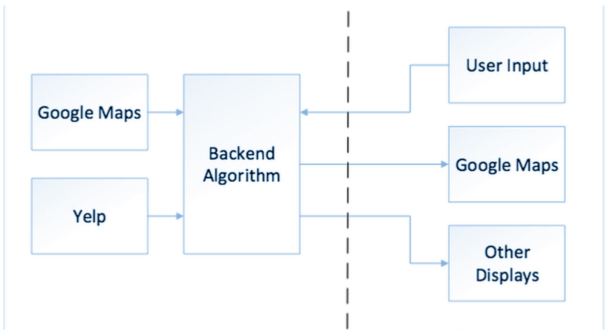
Figure 3: customized route

Another information we need to know from the user is his or her preferences. What attracts this person, and what the person tries to avoid? Then from the users selections, we will go to compare the similarity between the scattered points and the user preferences. Then we will further customize the route and make it special to this user.

4. DESIGN APPROACHES

4.1 Overview of the design

We used multiple tools and programming techniques to develop this application. Below is the high-level system architecture diagram for Tripster:



We built a customized Google Maps using Google Maps JavaScript API as our main front end display. Along with the map, all interactions and design are done purely in JavaScript (JQuery) and HTML. The front end is communicated with a Python program backedn server hosted in Google Apps API. It takes inputs from users at the front end, then it uses all this information to query routing information from Google Map API. After getting the routes, it breaks the routes into small "boxes" (we call the algorithm "boxing algorithm") With all the boxes, the back end queries attractions from Yelp with Yelp API. When it finally has all these information, it does a K-mean clustering to cluster attractions into small clusters along the route for easier viewing and selection. Then all of these information will be passed to front end to display.

4.2 Yelp Querying

In order to differentiate our website with existing route recommending websites, the incorporation of points of attractions is one of the keys to it. The yelp provides a relatively easy-to-use and flexible client API. The only tricky part to use the API is to set up the oauth authentication. While it's not very hard to use oauth to send a single query at one time, to query hundreds of queries simultaneously need to understand how the oauth works and decipher it. Since we need to obtain the businesses information around the whole trip, we need to send so many queries to the yelp server and get response from them as soon as possible. We finally use Google URL Fetch API to achieve the work.

One thing to mention about Yelp Querying is that since we are paralleling the query process so the number of queries does not affect the wait period of the query, however the Yelp has a daily limit per query and we actually hit that number once, so as will be discussed later, in order to do queries better, we will need to purchase the professional license of the Yelp API to avoid getting to the limit. There is also another interesting fact about information returned by Yelp API is that for a few days, we thought Yelp API's return results do not include the longitude and latitude information about the stores since it is not described in the API documentation as one of the data field. So we spend a lot of time searching for ways to convert from address to longitude and latitude info. When we finally found a 30GB database where we can download and access, we found out that the returned queries do actually include the information we need.

4.3 Boxing Algorithm

The default yelp querying method is searching by radius, however, this is not very appropriate in our setting. What we are trying to achieve is to search along a route with evenly separated interval. Also, we want to cover all of the businesses ranging within, say 25 miles, from the route. If we use the normal search query, it will be extremely hard to cover the whole area equally dense or we will waste too many queries (number limitation).

So we came up with the "boxing" algorithm. We first select via points equally spaced along the route. Initially the points are spaced relatively densely. Then we set one level of soft filtering that merge the squares that overlapping large areas together. So then we will have sparse and evenly spaced squares around the route. There will be a maximal width set to the square. So we can also guarantee that enough queries will be sent in order to no lose some of the top ranked businesses.

4.4 K-means algorithm

After the querying results are back, we then need to filter out the low-ranked results, as they will (by our assumption) generally be less interested to the road trippers. By now we have the first round of interesting data. However, if we are just going to display all of them to the users, it will likely to overwhelm the users as they will see so many points of interests scattered everywhere around the map. So we apply the k-means algorithm on the list of high-ranked businesses around the route. As a background introduction, k-means is a very classic clustering algorithm that measures the distance between points by some pre-defined metrics and then use an iterative approach to automatically assign all points to k clusters, where the distance between the cluster center and any point in the that cluster will be the smallest compared to all of other cluster centers.

Also, similar to the boxing algorithm, we need to impose some restrictions to this algorithm like non-overlapping and maximal area limit. With proper choice of parameters, the experimental results are actually quite satisfactory. More details of the results could be found in the "Use case demo" section.

4.5 Front end

As described in the first section, the front end is developed with JavaScript (along with JQuery, AJAX) for dynamic interactions, HTML and CSS for styling and Google Maps API JavaScript API for the main map display.

When you first loaded our page, you will see a nice video-backgrounded page to attract users. When people decide to try Tripster. They are mainly seeing a customized Google Maps with a theme of blueness and yellowness. We changed the color theme of the map so it aligns with theme of the application. On top of the maps, users will see a text input field to enter origin and destination. Underneath the input box, people can see the attraction selection bar developed with JQuery and ICONs from "FontAwesome". Upon receiving the clustering information from back-end, the interface will draw the clusters and display the transparent yellow clusters whose size and location is determined by the attractions inside them. Meantime, the map will be re-centered and re-zoomed so users are able to view all the information well. When users click the clusters, the view will be further zoomed in and see the location of each attraction shown as icons matching the icons in the preference selection bar. Clicking each attraction, a window showing semi-detailed store information will pop up on top of the icon. After viewing and searching, people can add their interested attractions and by the end users will receive a route connecting all of those inter-media points.

The challenges in front end is mainly caused by unfamiliarity of the interface programming. None of us has any experience with web front end development basically self taught us JavaScript (JQuery and so on) and HTML while doing this project. It was very frustrating at the beginning and it gets better later on. One particular problem we encountered was that one of the elements was not working, we could not figure out why and we did not receive any error messages. It eventually turned out that it was due to the incorrect order of including two high-level packages (one depends on another but we did not know about that). Also the nature of JavaScript is not the same as other languages (I should probably say web programming is different.) So a lot of simple tasks that can be achieved with ease in other languages are very hard and we sometimes want to achieve some functions that we did not even know how to Google or how to put them in a question sentence. But we received a lot of help from W3School and StackOverFlow/

The next challenge we found can mean either good or bad when it comes to programming. In web programming, each functionality people want to achieve, there are many different ways of doing it. They look the same, but they are still different in long run. This caused us a lot of problems since sometimes, two methods do not go along with each other well, so we had to pay attention to the details before we make decisions. But this situation gets better in the later stage when we know better about web programming.

The interface of this application is actually one of the most important part of the project and we spend way much more time developing it than I thought we would. We had a few different iterations from the initial design to the final stage and we have to admit the fact about developing web interface that we are never satisfied with the design. We always see something nice and always change the design. It takes a lot of time but luckily we are always making it look nicer. We have 3 major design changes and uncountable small modifications (colors, positions, and shapes of elements). Looking back at our first a few designs, we are very glad that we made those changes. We found it important to talk with multiple people to hear others' opinions about the design and learn about how to improve the design. We believe that interface will never be perfect, there was always room to improve.

Below is part of the revolution of our front end design. It was a pity that we did not have all of the intermediate steps saved, but you will have an idea about what is going on:

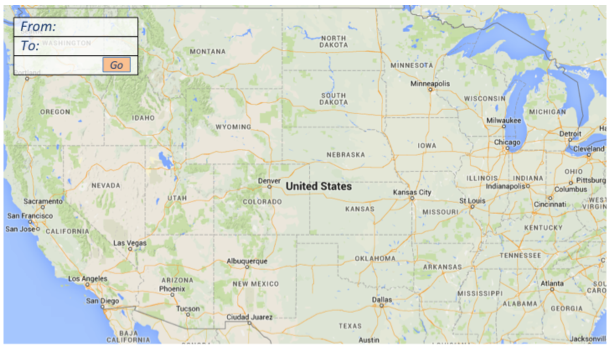


Figure 4: initial design: front page

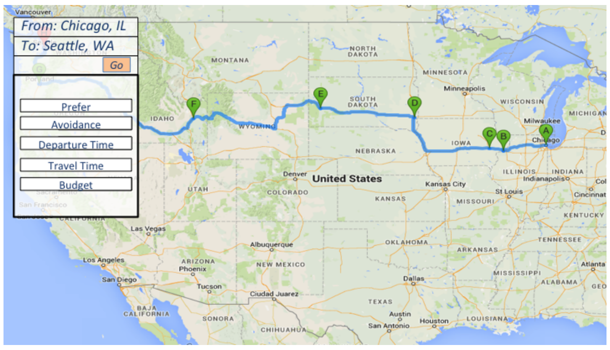


Figure 5: initial design: second page

As you can see from comparing this with our final design, we made a lot of changes during the course. Initially we wanted to use the same color as Google Maps, not only because to be close to Google but also because it was easy to implement. However, we changed this relatively quick in

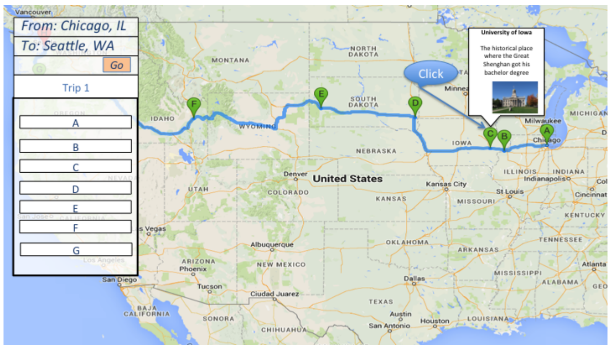


Figure 6: initial design: routing

the second iteration since we want to make us different from anyone and we did not like the color theme of Google.

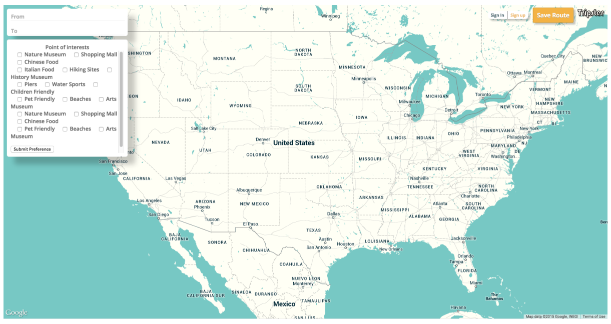


Figure 7: Second design: first page

This is quite similar to final design but lots of elements get changed. For example, the preference/attraction selection area was changed from the radio button selection inside Accordion to the icon selection bar. This switch is mainly because that we were taking E120 - Data Visualization from which we learned the importance to use icons to substitute massive text to make the visualization clearer.

5. USE CASE DEMO

Below is the demo of the use-case for the application with screen shots from the real application.

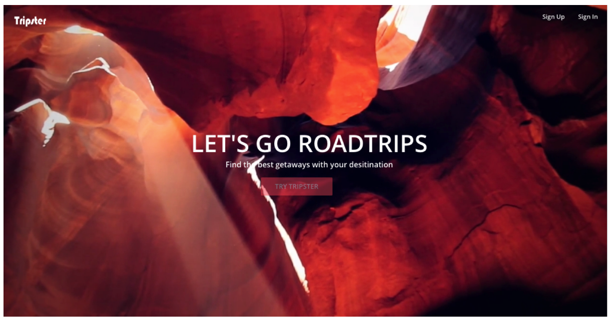


Figure 8: entry page

When the users enter the website, the figure above will be what they see first. This page acts as an inspiration page, which shows the purpose of the website and attracts people's

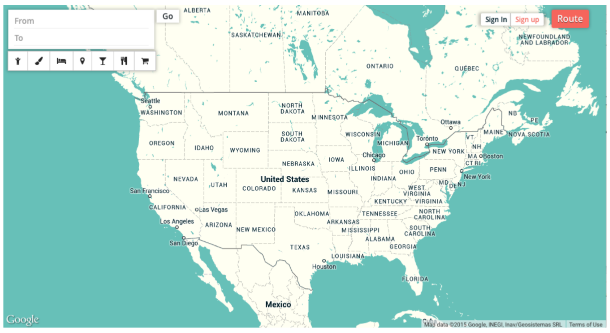


Figure 9: main page

attention. After the user selects the button of "Try Tripster", the main page will be loaded as above. The map is a embedded Google Map with special choice of color scheme. On the left hand side, there are an input field to enter the origin and destination and a dropdown group to select the user preferences. We get the idea of design of dropdown menu

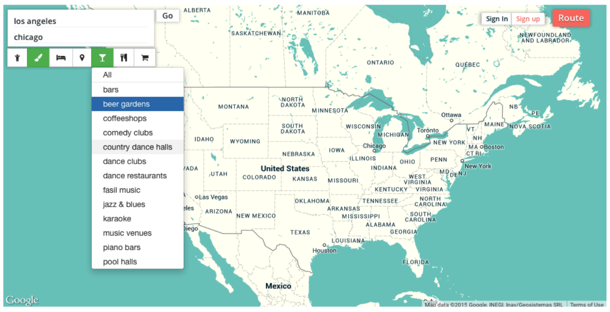


Figure 10: dropdown menu

from the website of "roadtrippers.com". They have done an excellent job in using the least space to display various of choices without losing any intuition. We follow the design principle but also have some minor changes. For example, the dropdown will display on hovering mode. This modification will make the user experience even more smoothly. After the user selects the preferences and also the targeting

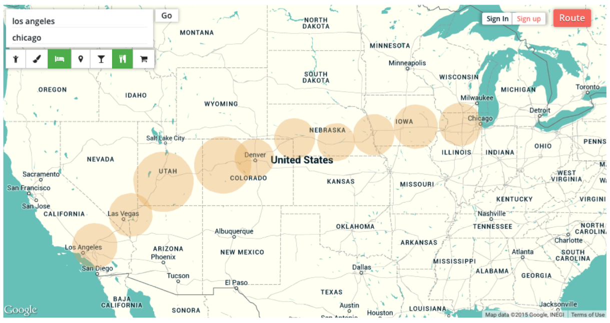


Figure 11: clusters of interesting points overview

point to go, a nicely organized overview of routes will be shown on the site. In this specific example, the user chooses to go from Los Angeles to Chicago. There are ten clusters shown on the web, indicating ten possible interesting stopping points to stop by. They are separated with reasonable space, so those will behave like different checkpoints. For every a couple hours of driving, the user (roadtripper) could enjoy a moment of leisure time.

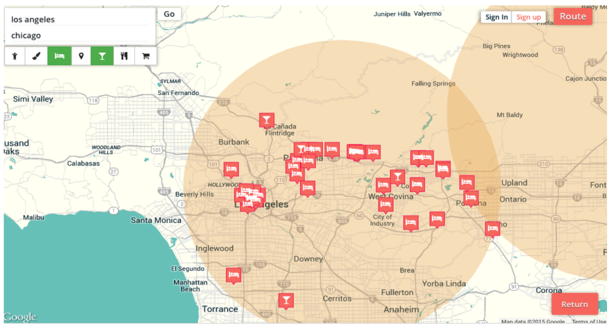
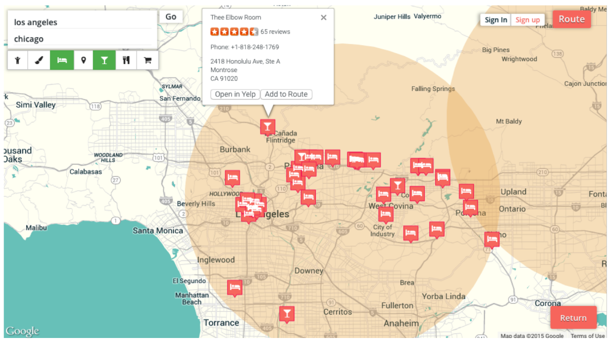


Figure 12: detailed view of a cluster

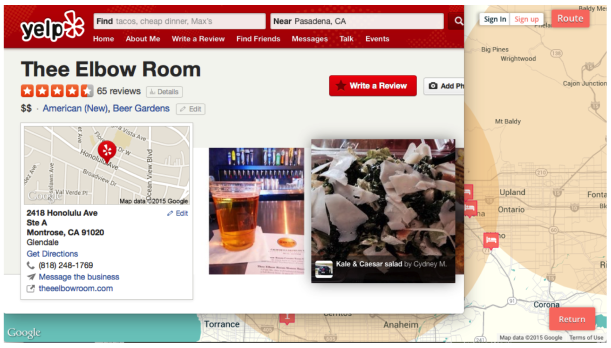
After the overview of clusters, the user may want to see what are the nice points to visit inside the cluster. So the user will be able to click one of the clusters and then a detailed cluster view will be presented. Basically all the interesting stopping by points within this region will be shown.



Thee Elbow Room
65 reviews
Phone: +1-818-248-1769
2418 Honolulu Ave, Ste A
Montrose
CA 91020
Open in Yelp | Add to Route

Figure 13: pop-up information about each store inside clusters

When users click on any of the icons/attractions. A window summarizing the attraction information including Yelp Review score and number, phone number, address and other options will show up on top of the small icon.



Thee Elbow Room
65 reviews
$$ · American (New), Beer Gardens
2418 Honolulu Ave
Ste A
Montrose, CA 91020
Glendale
Get Directions
(818) 248-1769
Message the business
theelbowroom.com
Kale & Caesar salad by Cydney M.

Figure 14: yelp pop-up

If users want to know more detailed information about the specific attraction, they can click on the "Open in Yelp" button to bring the Yelp page for the store. They can close the window anytime if they want to return back to Tripster.

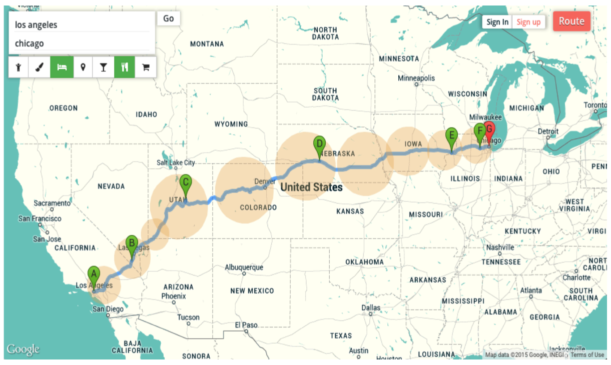


Figure 15: Finally, route!

After users have selected all the points of attractions that they want to stop through during the trip. All they do is to click on the right top corner "Route", this route will show up and users can enjoy their trip now.

6. KEY COMPETITIVE ADVANTAGES

Current there are a few other options for people to plan road trips. The first one is the conventional trip planning method and the second one is using other web applications as an assistant (Roadtripper for example). Comparing with them, Tripster has three major advantages:

First, Tripster offers the easiness to find and display all information on a nice web interface. With the help of the Tripster, users do not need to spend as much time as the conventional methods searching and Tripster is able to show a lot of interesting local attractions that are not known by tourists or even local residents.

Second, Tripster displays the completed but not overwhelming information to users. Unlike other road trip planning assisting websites which display all of the results to user at once, Tripster only returns top results and display them in clusters so users can easily select and view clusters without looking at 200 other attractions on the same page at the same time.

Third, this is our future feature but we have done a lot of foundation work to make it possible, Tripster will more input information. For example, the users can tell us types of attractions and accommodation he is looking for as well as users' time and financial budgets. Tripster can help the users to find best routes meeting all these requirements and maximizing users' utility during the road trip. This is extremely important for busy travelers and people who want to get hints about planning before spending time doing more researches. Another recommendation system is to recommend attractions and routes to users based on users' preferences, past history and specific behaviors. One particular example is we can link the account with users' Facebook accounts and access the user's liked photos of his friends. So if we happen to see the user's next queried route might go through a place where the user's friend has taken photo and he liked the photo. Things like this will make the system smarter and help people figure out what they want more. Moreover, if multiple "good" routes are found, we can assign a score to each of the route in different categories and let users to choose which one he or she would like to choose.

7. FUTURE WORK

To bring stable Tripster to the public with some competitive advantages, we still have some future work or future features to do or add:

First, we need to improve the query speed. Currently we are using free version of all APIs and servers, so we have lots of speed limitation or throughput limitation, so in order to launch Tripster, purchasing Pro licenses is very necessary. Another way to speed up query is to set up our own database to store all of the queried information to avoid repetitive query which is a waste of time and query bandwidth.

Second, Tripster is most likely to be used on desktop/laptop. However right after the trip has been planned, being able to access or the trip or even navigating on top of the application can help us gain an edge in competition. So in order to do that, we should develop a mobile application so that users can access the saved routes and navigate with our map during the trip so they do not need to switch between different devices.

Third, Tripser needs to have a login system so users can store their saved routes or previously planned routes in the data base. At the same time, we can learn users' preference and recommend them best routes matching their preferences. In order to do so, we need to set up a database to store users' information.

Fourth, this is something that will probably never reaches to the end: improve the GUI design. We have a good GUI already but there are many things that can be implemented to be better. For example, when loading the map, we can have better way to hide the latency. Now what we do is to use a transparent picture over the entire page, we can do it better. We can have the clusters show up one by one and have the color goes from light to dark until it is fully loaded to show the progress.

Fifth, we need to obtain more store information other than from Yelp. Another important feature we want to have is to display real-time closest open gas station information while they are driving with our navigation application on their smart device. I l learned this is very important from my personal experience: Once during my Iowa - Denver road trip while I realized that I need to refill at mid-night, I found a few gas stations nearby but when I stopped they were all closed. I was eventually told the nearest open gas station is 100+ miles away. I made it by turning off all electronics and drove at 40mph. I think it is very important to know these information on the road, especially when you are traveling with your family. So I hope Tripster could come up with the solution.

This is the extension of the last idea, we hope the routing system can be smarter while planning the route. For example, if Tripster finds the only gas station is far away from the regular route, Tripster can recommend user with alternative routes that can go through the gas station and refill before hitting the road.

Last, we have a lot of future features that we hope to include in the application. One particular example is that Facebook related described above. We hope our planning system can not only do what human beings can do faster, but also do what human beings are not able to or have not thought about doing.

8. ACKNOWLEDGEMENT

Professor Adam Wierman,
Google Maps JavaScript/Python API
Yelp API
Google Apps Engine
FontAwesome

9. REFERENCES

[1] Google Maps API USer Manual: https://developers.google.com/ma
[2] W3School: http://www.w3schools.com/
[3] StackOverFlow: http://stackoverflow.com/
[4] Yelp API: https://www.yelp.com/developers
[5] Roadtrippers: https://roadtrippers.com/
[6] TripIt: https://www.tripit.com/
[7] TripAdvisor: http://www.tripadvisor.com/