FineDyne: A Tool to Browse and Compare Restaurants

Dilan Ustek and Matthew Chun

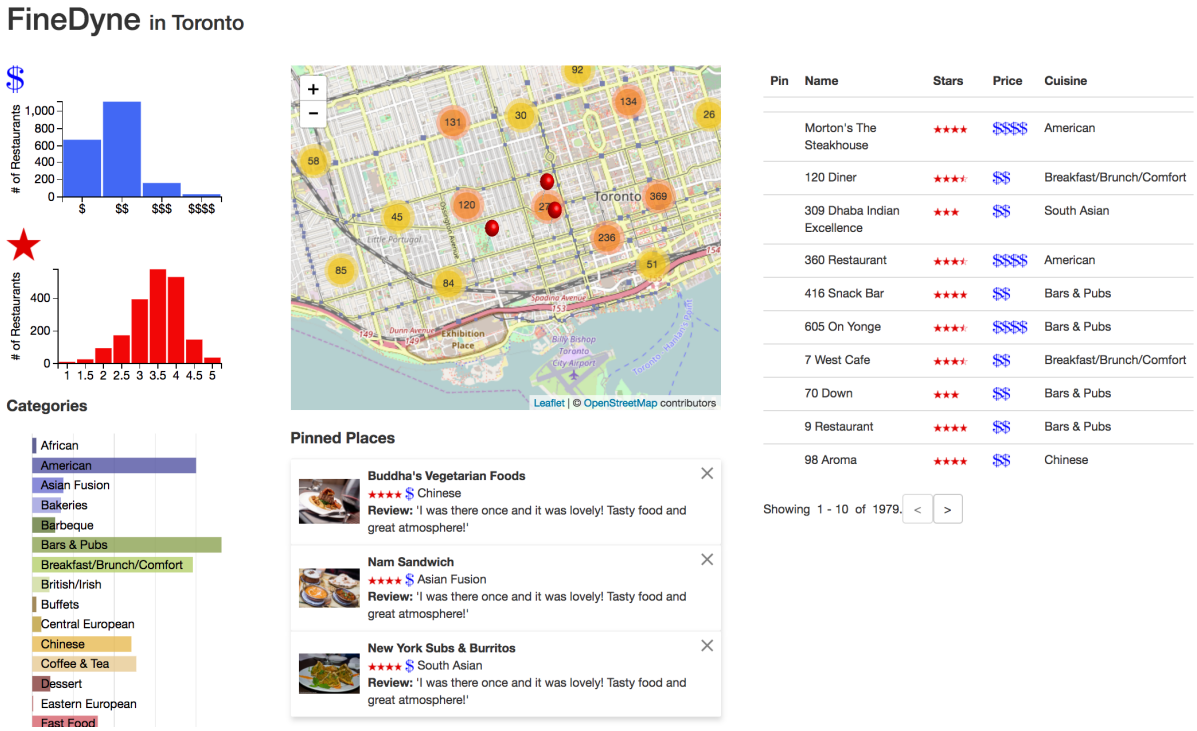

Fig. 1. The main overview of FineDyne with all settings at default. Pinned restaurants from past queries are intentionally still present.

Abstract—Deciding what and where to eat can be challenging due to the gaps in criteria known to finalize a dining decision. Current solutions such as Yelp not only withholds information to fill in these gaps, but also presents restaurant information in a narrow fashion that precludes more complex dining queries such as the comparison of two disparate cuisine categories. We introduce FineDyne, a web-based tool that leverages Yelp's own restaurant dataset and visualizes restaurant information in a much richer way. Specifically, we present the distributions of various dining criteria categories and link this with matching restaurants on a map view. Annotation features are implemented to allow users to save interesting restaurants of different criteria and compare them for a final more informed dining decision.

Index Terms—dining, food, visualization

1 Introduction

It can be surprisingly difficult to decide what and where to eat. Occasionally, we know exactly what we would like to eat. But it is more often the case that we don't have all of the necessary details to make the most informed dining decision. Yelp is a popular online service that provides restaurant information such as user reviews/ratings, descriptions, locations, and price details to assist in this common but important activity [8].

Yelp presents this information in a simple manner, in the form of a top list that matches criteria such as specific price ranges and cuisine types ordered by review rating and other metrics [12]. This list presentation can be appropriate for simple queries, but it is not conducive for slightly more complex but reasonable queries. For example, multiple restaurants cannot be compared based on different cuisines. Users currently can achieve this by using multiple web browser tabs. But this is not the most efficient way to compare restaurants of different cuisines in a holistic manner. Additionally, the distributions of different criteria cannot be known. As a result, users may use unpopular criteria combinations that may lead to less fruitful dining options.

To better support these complex queries, we introduce FineDyne, a tool which leverages Yelp's rich dataset of restaurant information that encourages a holistic and iterative way to compare various restaurants for a more informed dining decision. FineDyne achieves this by utilizing multiple views. A criteria view shows the distributions of various criteria such as the number of restaurants that fall under a specific price range, review quality rating, and cuisine type. This can be used to give users a guideline of interesting criteria that may yield rich results. A map view is used to geographically show restaurants of matching criteria to resolve decisions involving proximity. A comparison view provides an area for users to "pin" or save any candidate restaurants that they can use to compare against new queries. All of these views are linked together to reflect a filtering process of finding the best matching restaurants for a given combination of criteria.

Section 1.1 introduces the terminology used by Yelp that serves as the components of our criteria view. Section 1.2 will describe the notable limitations and problems with Yelp. Section 2 provides sum-

maries of related works that were used as inspirations for our chosen solution approach. In particular, we will describe works related to the usage of filters, scented widgets, and a multi-label faceted querying to encourage an iterative refinement of dining decisions. Section 3 will describe our tool's data, tasks, and their abstractions. Section 4 contains details of our solution and its justification as seen through the "what, why, how" framework by Munzner [15]. Section 5 will describe the libraries used by our solution and details regarding our experiences using these to implement our solution. Section 6 will address how our solution addresses the tasks described in Section 3. Section 7 will describe what we believe to be the strengths and limitations of our solution. Finally, Section 8 will conclude with a set of summarizing remarks and describe possible future directions for our tool.

1.1 Domain Terminology

While Yelp provides a myriad of criteria categories [9], we will be using the following categories based on our personal experiences and beliefs in regards to how most individuals would approach searching for a restaurant.

Price range will be used to differentiate restaurants on their average menu prices. Yelp defines this range in a discrete, ordered manner based on "price per person". With $ being the lowest price unit, consisting of under $10 per person, while the remaining 4 $ units represent increments of 20 to 30 dollars [11].

Review quality will be used to distinguish the restaurants on their overall quality. The review quality (stars) of a restaurant is rated by the users of Yelp and is represented by an ordered 5 point scale, using 0.5 increments. 1 star can be taken to mean that a restaurant was not well received by most users. On the other hand, 5 stars indicates general excellency from the users, generally with respect to their expectations. While numbers don't tell the entire story, review quality can be still useful as a guideline into assessing the general quality of a restaurant.

Yelp also provides restaurant information over 101 unique cuisine categories. How we chose to modify and use these cuisine categories will be discussed in Section 3.3.

1.2 Yelp Limitations

Currently, Yelp can be used effectively for simple restaurant queries such as finding the best ranked restaurants for a specific price range or neighborhood. But it does not fare well in answering more complex criteria.

Yelp presents the results of a given set of criteria in a ranked list. The restaurants on the list are ranked by their average review quality. Higher review quality restaurants are placed toward the top of this list. If a Yelp user desires to discover restaurants of some criteria based on popularity, then the list representation can be appropriate. However, the level of criteria detail given by users greatly affects the quality of interesting results in the list. Otherwise, a more random assortment of results can appear in the list. These results will likely be not very meaningful. An associated flaw is that users are not given information for better alternative criteria. Thus, it is difficult for users to know what the promising criteria combinations are in order to find potentially interesting results.

To an extent, Yelp can somewhat support the cases where users wish to compare different cuisine categories such as Japanese or Italian. For example, if searching for Japanese restaurants, a user can add more related cuisine categories such as ramen and sushi bars. However, it is not possible to add disparate categories such as Italian as seen in Figure 2. Yelp seems to operate on the assumption that only related cuisine categories are useful to display to the user. But with this restriction, it is difficult to compare against other types of cuisines. This necessitates users to start completely separate queries where the comparison of more diverse cuisine types must be done through external methods. For example, users can use multiple web browser tabs for each cuisine query, but this can be cognitively demanding.

2 Related Work

The related work covers different aspects of our solution. In particular, our solution was composed of facilitating a holistic, multi-faceted way to iteratively find the best result through filtering.

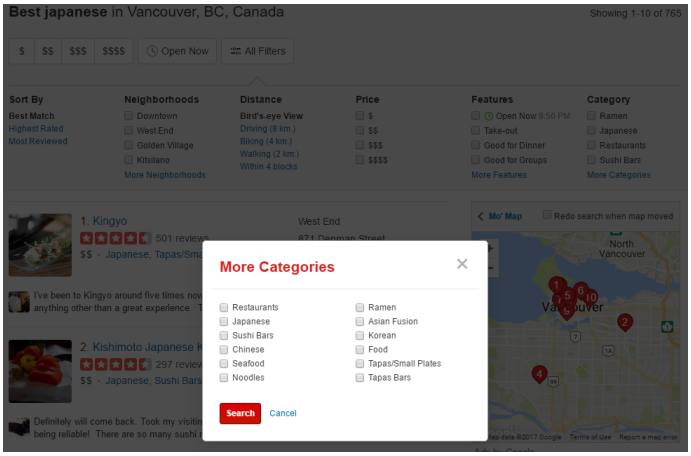

Fig. 2. Example of Yelp showing only related categories to original Japanese query. Disparate cuisine categories like Italian are not available.

2.1 Data Visualization Manipulation - Filters, Control Panels, and Widgets

A possible approach to control the visualization of data can be done through direct manipulation. For example, both VisExemplar [17] and dynamic queries [18] used this approach through the use of control panels, filters, and widgets where the users and the systems collaborated incrementally until the most effective possible visualization was created. Given the similarities in goals, we also can adapt the idea of a control panel with various options to filter the data in different ways until a satisfying visualization is reached.

However, given that our dataset can offer many possible types of queries, it can be difficult for a user to know where to begin their exploration process. Thus, it would be beneficial to guide users in this process, to know what might be interesting avenues of exploration.

2.2 Giving a start point for data exploration - Scented Widgets

Scented widgets [19] are visual navigation cues that can aid users in guiding and refining their exploration of a large dataset. For example, these cues can visually take the form of radio buttons with comments counts, or a slider with a histogram overlaid to show a distribution of the dataset, which can provide a perception of the cost and value of going down some path of information. In terms of our design, providing cues such as the distribution of popular restaurants split by different cuisine types could prove useful for users looking for a starting point for their restaurant decision making.

One issue with scented widgets, is that it may encourage users to search for mainly the more interesting or popular items in a dataset. While this is not a bad thing, this would undermine a more exploratory search process. Thus, an alternative approach would be to organize information into various different dimensions (facets) which can give the user a multiple angle approach to explore datasets without entering explicit search queries [13].

2.3 Presenting the data in a holistic manner - Multiple Facets

Flamenco [14] is one such example of a faceted search interface for images, that utilized multiple label querying to search for a specific subset of the larger facet space. For example, a facet may be food ingredients which could have the labels vegetables, breads, etc. Besides used as a querying method, Flamenco also had the notion of a query breadcrumb to help anchor users to their initial search criteria. This could especially be helpful in situations where many types of search queries may be employed, which could be useful for our design for

helping users understand what elements of their query led to their final result.

2.4 Filter and Focus - Subset on Distributions

It is likely that as users go about their dining decision process that they would need to filter and focus down to a particular subset of restaurants to compare. Vis-A-Ware [16] is a system for urban planning that allowed users to compare multiple candidate buildings on various visual occlusion types upon a citys landscape through a combination of a coordinated map view and table of histograms. The histograms denoted different distributions of these occlusion types for each candidate building or a particular real-life viewpoint. The users were encouraged to focus onto a certain subset of these distributions and filter onto them to see a smaller but specific set of candidate buildings/viewpoints that could be more readily inspected in the map. Thus for our design, we can encourage similar behavior by showing various criteria categories that restaurants could be viewed upon but offer mechanisms for refining the overall candidates to a smaller selection for easier comparison.

3 Data and Task Abstractions

The data and tasks used are related to the domain of using the Yelp service to find a restaurant for dining. We performed abstractions on both of these such that they could be analyzed more simply to help guide us to our final design solution.

3.1 Data Source

The data used for this project was provided by Yelp through their dataset challenge. The challenge encouraged developers to find new ways to use the data to help answer interesting domain questions [10]. Past example usages of this dataset have involved discovering cultural trends: Do Americans tend to eat out later than Germans? Does a restaurants popularity really have to do with their location? Is a Chinese restaurant also really Szechuan or Hunan style? Multiple datasets such as those with user restaurant check-in details, restaurant reviews, restaurant tips, and user reviews were provided. We take a more end-user approach with these datasets.

3.2 Data Domain

Based on Section 1.1's domain criteria, we used the provided restaurant review dataset. After converting the dataset from the default JSON format to a CSV format using the provided conversion python script, we found that the data contained the following in a table format:

- Restaurant names
- Location information for each restaurant eg. longitude/latitude, address, neighborhood
- Array of all associated cuisine categories of a restaurant
- Array of all extra meta information eg. Alcohol serving status, Credit Card acceptance, price range, etc.
- Operating Hours
- Unique business ID
- City of operation eg. Edinburgh (UK), Toronto (Canada), etc.

From the cities provided, we chose to focus on Toronto out of patriotic interest, where we looked at all rows of the dataset pertaining to the review quality and price range of a given restaurant. More details regarding the 101 unique cuisines and the 2203 restaurants identified in the dataset and how we aggregated them for our solution will be discussed in the next section.

3.3 Data Abstraction

We retained the data table format but simplified it for our solution requirements. In particular, our data table had the following:

- Each row represented a single restaurant (categorical) with an associated column for restaurant names
- Columns for price range and review quality (ordered)
- Columns for location information eg. longitude/latitude (quantitative), address (categorical), postal code (categorical), neighborhood (categorical)
- Column for unique business ID (categorical) for later data cleaning of duplicate entries
- Column for cuisine type of each restaurant (categorical)

101 unique cuisine categories were identified in the dataset. For the purposes of scalability with our final idioms, the cuisine categories had to be aggregated. What these idioms entailed will be discussed in Section 5. Many of the cuisines were "binned" to more generic but representative cuisine types. For example, the aggregated cuisine "Southeast Asian" would include individual cuisines such as Thai, Vietnamese, and Filipino. Generally speaking, we used the heuristic of geographic/semantic familiarity with the aggregated cuisine categories, where the goal was to identify if the aggregation could be used in place of a specific cuisine type in question. Occasionally, some cuisines were standalone such as "Gluten-Free" to cover special edge cases of notable distinctiveness that would be lost in aggregation. While this approach assisted in scalability, there were notable trade-offs. These will be discussed in Section 7. Please refer to Table 1 and 2 to see how we aggregated all of the Yelp cuisines.

Table 1: All aggregated cuisines and their individual cuisine categories that compose them. Aggregated cuisines are on the left side.

Aggregated Cuisine Table	
African	African, Ethiopian, South African
American	American (New), American (Traditional), Canadian (New), Hawaiian, Tex-Mex, Steakhouses, Burgers
Barbeque	Barbeque, Southern
Breakfast/Brunch/Comfort	Diners, Soul Food, Comfort Food, Breakfast & Brunch
Fast Food	Fast Food, Hot Dogs
Bars & Pubs	Chicken Wings, Bars, Beer, Beer Bar, Dive Bars, Pubs, Sports Bars, Tapas Bars, Wine & Spirits, Wine Bars, Poutineries
Sandwiches/Soup	Cheesesteaks, Sandwiches, Soup
Bakeries	Bakeries, Donuts, Bagels
Latin American	Brazilian, Cajun&Creole, Caribbean, Colombian, Cuban, Latin American, Peruvian, Salvadoran, Mexican
Dessert	Creperies, Desserts, Ice cream & Frozen Yogurt
British/Irish	British, Fish & Chips, Irish
Central European	Czech, German, Polish, Hungarian
Eastern European	Ukrainian, Russian
Mediterranean	Greek, Iberian, Spanish, Mediterranean, Moroccan, Portuguese
Coffee & Tea	Cafes, Coffee & Tea, Tea Rooms
Southeast Asian	Indian, Himalayan/Nepalese, Sri Lankan, Pakistani
Chinese	Chinese, Dim Sum, Hot Pot, Taiwanese

Table 2: All aggregated cuisines and their individual cuisine categories that compose them. Aggregated cuisines are on the left side. *Cont.*

Aggregated Cuisine Table Cont.	
Middle Eastern	Middle Eastern, Donairs, Falafel, Lebanese, Persian/Iranian, Turkish
Gluten-Free	Gluten-Free
Vegetarian/Vegan	Vegetarian, Vegan
Halal/Kosher	Halal, Kosher
French/Belgian	French, Belgian, Fondue
Italian	Italian, Pizza
Japanese/Korean	Japanese, Ramen, Sushi Bar, Korean
Seafood	Seafood
Buffets	Buffets
Asian Fusion	Asian Fusion

3.4 Task Domain

To represent the possible tasks that our tool can support, we present the following task examples of fictional users to represent the variety of requirements that our tool should accommodate. These examples were formulated from the designers and their own usage walkthroughs of the current Yelp website and the issues that were discovered.

Katie the birthday planner - Katie wants to plan her best friend's birthday with some of their closest friends invited. She knows for sure that her best friend is a huge Japanese fan but the other friends are more keen are Mexican. While these are important, it's also important to find a place that has great after party options nearby, such as a bar. Given that this is a special occasion, Katie is willing to spend more, upwards of a 45 dollars budget per person given that the restaurant is of higher quality (4 stars and up).

Analysis - The above task example is a case where the individual knows all of the criteria. It's simply a matter of finding the set of restaurants that best fits these requirements. For Katie to achieve this task currently, it is entirely possible to do it within the current Yelp website. She must look up separate queries for each interested restaurant type and use the price range filter. A caveat is that she must keep track of the bar locations with each possible Japanese or Mexican place that she is considering, which through a web browser would necessitate switching tabs or using multiple windows for comparison.

Andy the company party planner - Andy is the engineering manager of a mid-sized company in downtown Toronto. It is his responsibility to plan a project launch party for his team after work. He wants the party location to be at a restaurant close to the office, and especially would like great dessert options nearby (a cake is necessary for a celebration after all). In terms of the main course meal, he is much more flexible on this, as long as it's a reasonable price for the company to reimburse.

Analysis - This task example is a situation where only some of the criteria are known, but in a very loose manner. For Andy to achieve this using the current Yelp website would be a challenge. In Andy's case, he needs to be able to see nearby dessert places and a general set of "good" restaurants. The narrow focused searching of Yelp is a hindrance, Andy would have to first look up dessert places, then do a separate search for generally good restaurants nearby the office. Looking for a good restaurant may take him a bit longer, as he only has price as a general constraint. It wouldn't be possible for him to gauge the cuisine options nearby that might fall under a specific price range. If he could then this can be advantageous in that he can see if many restaurants nearby are of a specific cuisine, then it is likely that a specific popular cuisine maybe the "safe" choice to use for the party.

Sally the tourist - Sally is a tourist that is new to Toronto. She

has a list of some notable neighborhoods that she would like to visit in her limited time. She is very keen on finding the higher quality restaurants with no limits on the types of food they serve. She simply wants the best of each place that she will visit.

Analysis - This task example has the least amount of limits on the criteria required for a restaurant. Similarly to Andy's case to find simply "good" restaurants, Yelp will present Sally with a list of highly reviewed restaurants in some ranked order even when no price range is specified. But there is no way for Sally to know the exact distribution of restaurants of different cuisines that also qualify as highly rated. Yelp's ranking algorithm would "hide" this distribution which would lead to a lesser informed dining decision. For example, if Sally could see that one of her neighborhoods was dominant in offering high quality Indian food versus that of Thai food, she maybe more inclined to go with one rather than the other.

3.5 Task Abstraction

For all 3 of our task examples, all of the users wish to discover a set of restaurants that matches their specific criteria. The level of specificity in the criteria differed among all 3 examples, where some aspects such as cuisine or price range was known, but the resulting target restaurants were not known. Thus, all 3 individuals would have to browse for qualifying restaurants that matched their requirements.

Many of the task examples placed constraints on finding nearby restaurants in addition to the main ones to be found. Thus, the ability to compare restaurants based on a specific criteria is important to support.

Finally, it was implied that being able to see an overview of distributions may have led to more informed dining decisions in the cases of Andy and Sally. There is something to be said in seeing the number of restaurants that meet a requirement. Similar to how Yelp currently offers seeing the number of reviews in addition to the average review quality, being able to see the distribution of criteria categories such as price range, and cuisine type could be very informative for these users.

4 Solution

In Figure 1, we present FineDyne, which is composed of 3 main views that are linked together to support an iterative and holistic dining decision making process. In the very centre of the interface, is the map view that will be the main area where criteria matching restaurants will be populated. On the right of the map view is an associated list of matching results that is tied to the view. Any restaurants that match the current criteria and is located in the current map region will be displayed in this list. The purpose of this list is to support quick scanning of criteria results in situations where many restaurant markers overlap.

The criteria view on the left side of Figure 1 is used to begin the search process. A series of barcharts are used to represent what we believe to be the most useful Yelp criteria. The criteria involved review quality (stars), the associated cuisine type based on our aggregation, and the price range of a restaurant ($ sign). The purpose of this view is to give users a sense of where to begin their search process by providing the distributions of these criteria categories, which could give hints of what the most promising criteria combinations could be in order to find their final restaurant. The individual bars of these charts can be selected through a click, which would filter the map views results based on the selected bars of interest. To support an iterative decision making process, these bars can be freely toggled on/off to change the resulting map view results at any time as seen in Figure 3 and Figure 4.

Selected bars would be saturated while de-selected bars would be de-saturated. To support tasks involving location based decisions eg. Which restaurant is closer to X, the criteria view changes its distribution in accordance with the current zoom level of the map view.

Early in our design process, alternative ways to filter results were considered. Our initial approach in criteria selection was quite simple. A series of checkboxes were used to select the desired price, review quality, and cuisine choices, with the results then presented on the map view and another chart area. In this chart area, matching restaurants would be positioned along the axes of price and review quality. The

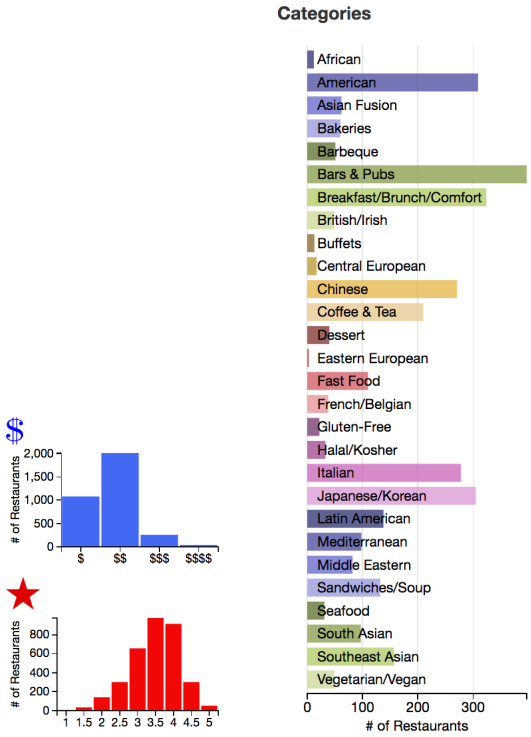

Fig. 3. Criteria view barcharts can be freely toggled on/off to select specific subsets of criteria areas of interest. All bars are selected in this example.

restaurants would be shown in cards to indicate how different cuisine restaurants stood relative to each other along these axes as seen in Figure 5. But due to the lack of scalability to accommodate a high number of results, this approach was quickly discounted.

Another early design consideration for the criteria view utilized a 4x5 matrix grid. Rows represented the 4 price range categories, and the columns represented the 5 star review quality units. Each element or a range of elements of this matrix could be toggled on/off to accommodate a combination of different price/quality criteria eg. "4 star restaurants in a $$ and $$$ range". Please refer to Figure 6 to see this in action. The flaw with this approach was that it obscured the promise of a potential criteria combination, where the results may not be very fruitful. In other words, users should be able to know beforehand whether a particular criteria combination would be interesting to embark upon, in order to not waste their time

As in seen in Figure 7, a stacked barchart view was considered at one point to display the proportion of review quality stars for each cuisine category for our criteria view. The stacks were thought to serve as a good quick way to gauge the quality distribution of a single cuisine type in question. However, due to price category information unable to be coded, we decided that using separate barcharts to represent the criteria categories were more appropriate. Breadcrumbing was also planned initially, but this was discarded as the selected bars in the current criteria view served a similar purpose.

As mentioned previously, the map views role is to display the results of the criteria selected in the criteria view. Markers represented as blue balloon glyphs are used to indicate the appropriate criteria matching restaurants. Clicking or hovering over the marker will show a pop-up window (detail on demand) with summarized restaurant information such as a 1 liner review, an image, and price/review quality. This information can be used to judge if the restaurant should be pinned to the comparison view area for later consideration. The pinned restaurants are represented as physical red pins to visually distinguish themselves from the remaining markers on the map. In practice, we found this to serve their purpose of being distinct and easy to find due to

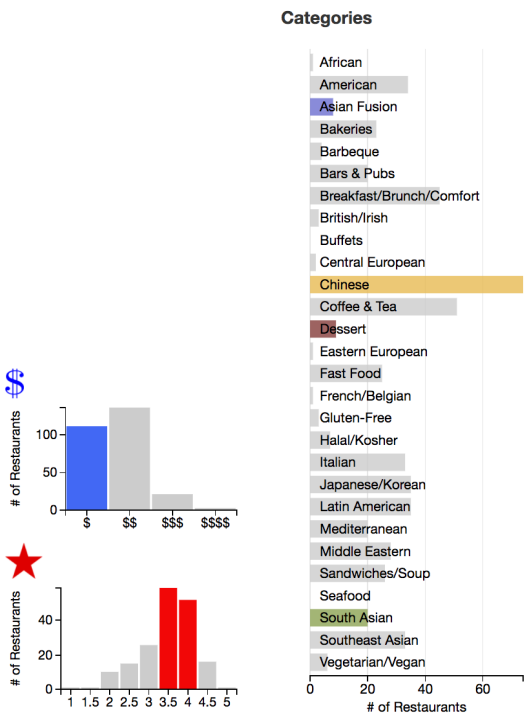

Fig. 4. Criteria view barcharts can be freely toggled on/off to select specific subsets of criteria areas of interest. Only some bars are selected while the remaining are deselected in this example.

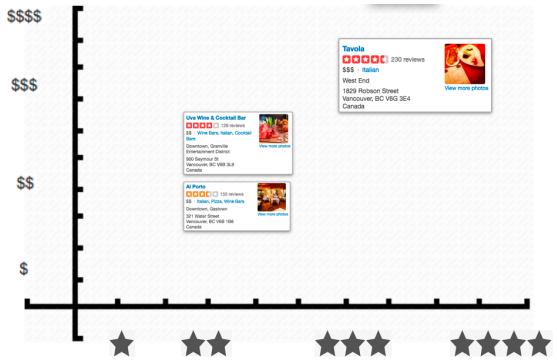

Fig. 5. Early design consideration for criteria view. Matching restaurants are shown in cards relative to each other along different criteria.

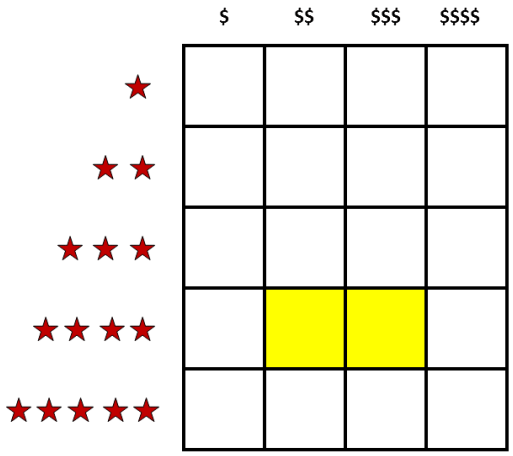

Fig. 6. Early design consideration for criteria view. Specific grid elements can be selected to investigate specific criteria ranges.

the lighter hues used by the map implementation. Our map implementation provided a way to aggregate markers based on the current zoom level of the map. We felt the aggregation was appropriate as the unique markers from the aggregated clusters would only be visible at the closest zoom level. This was appropriate, as this was where users would likely be performing any proximity decisions. Secondly, the aggregation was useful to quickly get an overview of areas to find matching criteria restaurants. Please refer to Figure 8 for an overview of the map view. In cases where large amounts of restaurant markers would appear on the map view, we created a list that displays all of the restaurants in the current map region in an easy to read table. The table's row contained the restaurant entry, with the columns containing criteria category information such as restaurant name, price range, review quality stars, and cuisine category. Each of these columns could be sorted by ascending/descending order to quickly find and compare restaurants on a specific criteria category.

The comparison view is located on the bottom of the map view, and is dedicated towards the displaying of pinned restaurants. A "pinned" restaurant denotes a potentially interesting restaurant that a user would like to use later for more final dining comparisons. Pinned restaurants persist across different sessions of using FineDyne, even when users decide to investigate different criteria combinations. Thus, this supports an iterative decision making process, where users gradually compile a list of candidate restaurants for a more holistic dining decision at the end. Hovering over the pinned restaurant will link highlight the associated map marker in red. This was done in order to support tasks involving proximity among the pinned restaurants. Please refer to Figure 9 for an example. We decided to place the comparison view on the bottom of the map view in order to encourage its importance. It is very literally at the centre of the user's visual attention, which should make it salient for the more important comparison tasks later on.

5 Implementation

5.1 Data Parsing

With over 2203 restaurants that often repeated themselves for each of the 101 unique cuisine types recognized, the restaurant review dataset had to be parsed down into our interested criteria information. This was achieved through a combination of R [6] and Trifacta Wrangler [7]. Both of these tools were suitable for manipulating tabular data.

R was used to breakup the massive restaurant review dataset into just the entries related to Toronto. This was achieved by using the R library package data.table which allowed for SQL-like queries. A query was made to extract just the rows that specified a restaurant was from Toronto versus another city in the dataset. The Toronto entries were then saved to a single CSV.

The next step was to discover the unique cuisine types of Toronto. Annoyingly, the original format of the dataset compiled all possible cuisine types for a restaurant into an array entry. This was likely due to the possibility of having a restaurant fall under multiple cuisine types eg. Japanese and Korean, Indian and Pakistani, etc. Trifacta Wrangler was used to break the main categories column of the dataset, that held these cuisine arrays, into split columns that contained a single cuisine category each. The splitting itself was achieved through a built-in function called "split" that would split a column based on a provided string pattern such as comma in our case. Once the split columns existed, R could now be used to count all unique instances of each cuisine type mentioned in each of the split category columns. Rs built-in table functions worked well to give a summary output of unique cuisine types. We used this information to inform the final set of aggregated

Fig. 7. Early design consideration for criteria view. Can select subset of criteria ranges to investigate.

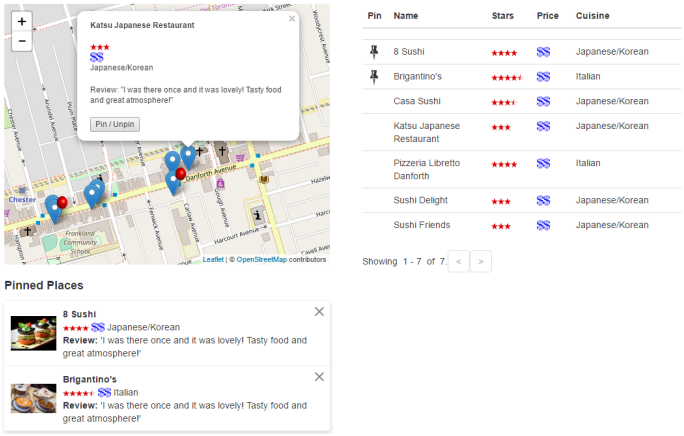

Fig. 8. The map view in action. Here we demonstrate pinning and the summary list view corresponding to the current region of interest.

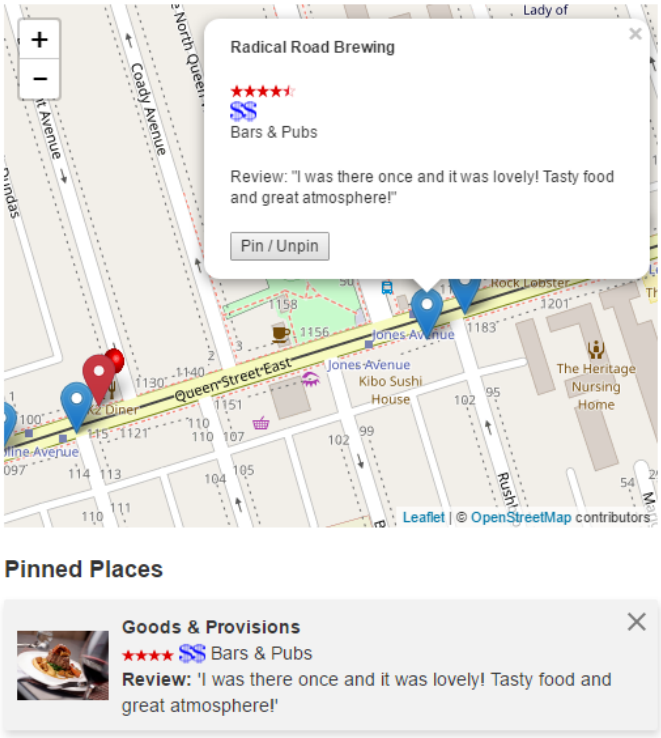

Fig. 9. An example of compare view demonstrating linked highlighting with pinned restaurants. Highlighting is shown in red on the map view.

cuisine categories.

With the cuisine categories now known, we had to then create separate CSV files that contained just the restaurant entries that pertained to a specific cuisine such as Japanese. To do this, first R and the data.table package was used to set a key on each category split column, to look for the cuisine type in question along it.

R's data.table package was used to set a "key" on each split category column. All found entries that matched the current key value (eg. Japanese) for the split column was then added to a R vector. Then it was a matter of writing each vector into their own CSV files. These separate CSV files represented all of the found cuisine entries for a specific cuisine type in Toronto. Trifacta Wrangler was then used to import all of these CSV files, and union them. Again, Yelp had annoyingly hid the required price category information into an array that contained a dump of meta-restaurant information such as credit card friendliness, etc. Trifacta Wrangler's split function was used to segregate the price range information into its own column based on a specific string pattern (eg. "RestaurantsPriceRange2:"). Rows of the dataset that were missing information such as location were dropped. We also dropped columns that represented extraneous data from the arrays that were split. Other minor actions involved rearranging the order of columns for readability purposes.

This process of creating cuisine specific CSV files took about 6 minutes for each of the 101 cuisine types. The process was semi-automated via 2 scripts, one for R, and the other for Trifacta Wrangler.

To create the aggregated cuisine CSV files, Trifacta Wranglers union function was again used. When all of the CSV files required were created, it was a simple matter of running Trifacta Wranglers union function on all of the aggregated cuisine CSV files. In this final CSV file, we also removed any non-active businesses to simplify the size of the CSV file to account for possible performance issues. We also removed duplicate entries based on the unique business ID described. Duplicates occurred due to our new formatting of having a restaurant of a specific cuisine type to be its own row, as we removed the multiple tagging array of cuisines that the original dataset employed.

5.2 Tools for Front-End

FineDyne is implemented using a combination of different web-based tools. The front end was developed using dc.js [3], a JavaScript library that utilizes d3.js [2] in order to render a variety of large multi-dimensional charts in a CSS-friendly SVG format. The charts themselves are driven by the provided data and is very reactive, making it appropriate for facilitating the interaction of FineDyne.

Another important benefit of using dc.js was its compatibility with Crossfilter [1], a JavaScript library that supports coordinated views with huge datasets. It manages to achieve a very quick performance (30ms) with large datasets due to its usage of incremental filtering and reducing approaches to these sorts of datasets. The map view of the tool was implemented using dc.leaflet [4], a variant of Leaflet [5] designed to work with dc.js and allows for interactive map usage. Dc.leaflet allowed for dc.js charts to be linked to the maps rendered by Leaflet.

5.3 FineDyne Components

We loaded the data using the d3.csv function which also contained references to the drawmarkerselect function, which was used to render the necessary views (refer to web/js/fdComponents.js for more details).

FineDyne's views were rendered by loading our aggregated dataset using the d3.csv function. The actual rendering of the views was performed through the "drawmarkerselect" function. This function was called at the same time our data was loaded. Please refer to the project source code at *web/js/fdComponents.js* for more details.

We implemented the price range and star quality distributions with dc.barChart and the category types were represented with dc.rowChart. The map was a dc-leaflet.markerChart. The markers themselves are displayed in accordance to a named restaurants latitude and longitude information via a location accessor function provided through dc.leaflet.js. The list view was a dc.dataTable. These components were all linked by using the same Crossfilter.

The dc.BarChart and dc.rowChart components were used to imple-

ment our price range and review quality distributions for the criteria view. The map view composed of a dc-leaflet.markerChart object. The map view's markers were displayed in accordance to a named restaurant's latitude and longitude information. This location information was obtained through an dc.leaflet.js accessor function. The map view's list was a dc.dataTable object. The information shared among these different views was achieved by using Crossfilter.

By providing different dimensions to the same Crossfilter, charts can be efficiently linked together, filtering and showing the same data, but grouping them in different ways. For example, the star bar chart, and the price bar chart show the same restaurants, but they group they using different "dimensions": one uses stars as the dimension, the other uses price range.

Charts can be efficiently linked together by filtering and showing the same data across different charts. This can be achieved by providing different dimensions to the same Crossfilter object. For example, the review quality and price range barchart can show the same restaurants, but they "group" the restaurants through different dimensions. While the review quality barchart provides "stars" as one dimension, the price range barchart uses "price range". Reduce functions from the Crossfilter library are used to form an individual bar's height in any of our barcharts. For example, the reduce functions count the records of each matching cuisine type in order to compose an "aggregated" group that will act as the individual bar.

Dc.js makes it easy to create charts out-of-box by giving it the necessary options and parameters. Charts can be customized to have certain properties such as "elastic" x or y axes that can rescale themselves depending on the range of the data, certain label formats, and clustering of data in the dc.leaflet.js map object.

Using Jquery, we provided users with extra options such as making the map view's list columns sortable when a user clicks on the column labels. We used d3.js to customize the charts even further by setting color palettes, ordering sorted data, and implementing the pagination of the list view.

5.4 Pinning

The compare view of FineDyne contained interesting challenges. A core design idea of ours was to allow users to be able to pin restaurants of interest in order to compare them later. However, there was no dc.js chart that could accomplish this feat. Despite the simple idea, this was not trivial to implement as not only should users be able to pin and unpin restaurants from the map, but the same actions had to be allowed from the map view's list as well.

The list view was generated by dc.js automatically using the same Crossfilter as other charts. This meant that we needed to generate each row in the table with an JavaScript onclick function. When clicked, this would send the function handling the clicked restaurant's data to a function that would deal with pinning. Additionally, the map view's pop-up windows were generated and binded to each marker element individually on load, and needed to call the same pinning function when a user pinned a restaurant from the map view's pop-up window.

Users can "pin" a restaurant by clicking on the row in the map view's list or by clicking the "Pin/Unpin" button on its pop-up window. The function for pinning, called "togglePin" takes the businessId of a restaurant, and keeps track of pinned restaurants using an object data structure. A pin icon appears on the row that corresponds to the restaurant on the list view when a restaurant is pinned and it disappears when it is unpinned.

5.5 Linked Highlighting

Another important contribution of our tool is that users can compare the proximity of different pinned restaurants. The easiest way to do this is to highlight the location of a pinned restaurant when a user hovers over it on the map view's list or from the compare view. However, as we generate the markers on load, when users hover on the list view, it is hard to change anything about the marker. To make it work, we would need new instances of the marker in blue for the default look, and in red for the hovered look which would cause memory performance issues.

To avoid memory issues, we created a function that would create a red version of the blue map marker that would be inserted on the markerchart object at the exact location of the original blue marker. This function would be called when the map view's list element or compare view's element was hovered upon. The end effect is that when users hover over a certain pinned restaurant item, there is an impression that the marker is red to highlight its location on the map view. When the user clicks on the list restaurant item, they pin the restaurant onto the map via a red pin on top of the marker. This acts as confirmation of a successful pinning action. Even when users are no longer actively hovering over the pinned restaurant list element, the pin will persist on the list to indicate pinning is still active as well.

5.6 Disadvantages of dc.js

As we mentioned above, there are many advantages of using dc.js since it provides optimized out-of-the-box interactive charts. However, it is important to note the disadvantages in using such a powerful library.

The main disadvantage we experienced was that dc.js makes it hard for the developer to manipulate and customize the low level details of the charts. The developer needs to know a lot of the syntax and capabilities of such a library in order to use them well. As a simple example, a row chart in dc.js does not have a function for labeling axes. Adding this is not as simple as the lower level and more developed library, d3.js. For this case, we needed to write a function (addXLabel) to manually insert a x-label to the categories chart.

6 Results

In this section, we will present how FineDyne can be used for each of the user types and their task examples described in Section 3.4. Specifically, we will describe how FineDyne can accommodate different levels of restaurant discovery based on the level of detailed criteria a user possesses.

6.1 Usage Scenario: Katie the birthday planner

Katie, who needs to plan a birthday party, opens up the new FineDyne website. After the website loads, she first looks to the barcharts that shows the distributions of her desired criteria . Since she is interested in only comparing Japanese and Mexican places, she selects them to be the main filters by clicking on them in the cuisine category barcharts. The website then updates by showing the distribution of Japanese and Mexican restaurants on the map, which ends up removing quite a bit of the restaurant markers.

Next, since she has a 45 dollar budget and would like generally high quality restaurants, she clicks on the 3 dollar sign price range bar in the price barchart, which covers 31 to 60 dollars per person, She also clicks on bars representing 4 stars, as she desires a high quality range of restaurants to view. The map again updates accordingly, reducing the number of markers down further.

In terms of location, somewhere fun is important, so she decides to zoom in on the map to the main Downtown area. From her previous actions, she can see quite a few possible options at this point. She sees from the summary list on the right of the map, that her current chosen area shows Yuzu No Hana and Los Colibris, both great Japanese and Mexican places that she has heard good things about from her sister. She decides to pin both of them for later consideration.

But her main constraint to solve now is to find a good bar nearby for a decent after party. She goes back to the cuisine category barchart, and deselects the Japanese and Mexican options by clicking on them. She then selects the Bars option by clicking on it. The map updates accordingly, and she can see in her current zoom level on the map, that a couple of bars appear. Using the summary list again, she can see a decent bar called Shangri-La.

But she wants to now compare the bar in relation to her previously pinned restaurants, where she will pick the restaurant based on its proximity to the bar, as she would like to minimize commute time to have more time for the partying. She hovers her mouse cursor over the pinned restaurants on the bottom of the map, and can see them linked back to the map in the form of red markers. She sees that Yuzu

No Hana is closer to the bar, and decides to use it for the main dinner. She jots down the name of the restaurant and bar for reservation later. Please refer to Figure 10 to see Katie's actions using FineDyne.

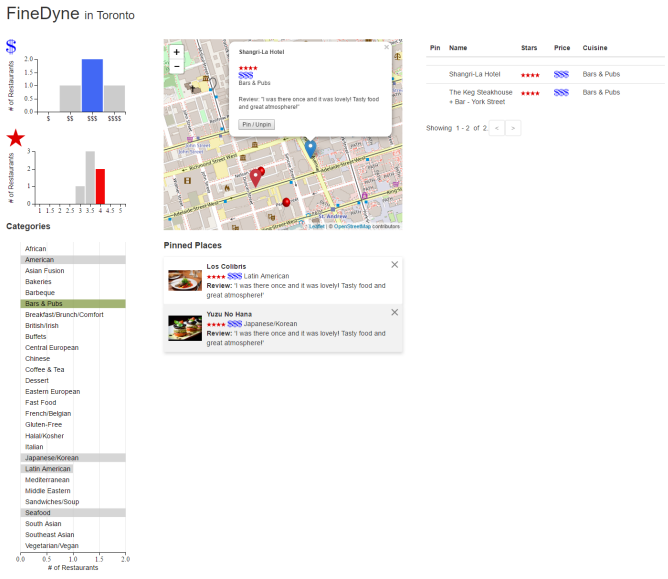

Fig. 10. Katie planning a birthday. She makes the final decision based on the Japanese restaurants with closer proximity to the Shangri-La bar (linked highlighting in red).

6.2 Usage Scenario: Andy the company planner

Andy, who is responsible for planning his company party, opens up the FineDyne website. Since his only hard constraint is to find a decent dessert place near his office, he first selects the Desserts cuisine category bar in the cuisine category charts. The map then updates with the matching results. He then zooms in to his office location on the map to see the matching dessert places nearby the office. From the summary list near the map, he can see quite a few dessert places nearby. As quality is important, Andy selects the 4 star and up bars in the review quality chart to filter out the matching dessert places. He sees a promising one on the summary list, and decides to pin it for the basis of the next step.

The next step now involves finding a restaurant to host the main dinner. Unlike the dessert place, Andy is much more flexible in terms of review quality, as price is the bigger concern. He selects the medium 2 dollar price range in the price chart, which updates the review quality chart distribution. He can see that there is a 5 star restaurant for the chosen price range and decides that this will be the place to host the dinner. He selects the 5 star bar to filter out the other non-relevant quality options. Hovering over the matching marker reveals that the restaurant is of reputable quality based on the 1-liner review. Andy thus decides to host the dinner at this restaurant.

6.3 Usage Scenario: Sally the tourist

Sally is a tourist new to Toronto. In her hotel room, she decides to do some research on finding the best restaurants for each of the neighborhoods that she plans to visit. She opens up her laptop and points her web browser to the FineDyne website.

Since Sally is keen on finding the higher quality places with no particular constraints on the types of food they serve (she is quite open minded), she decides to select 4 stars and up in the review quality chart. She leaves the price ranges at their default selection of all, as shes does not want to lose out on potentially interesting restaurants due to pricing concerns. She also leaves the cuisine chart alone, by leaving all of the cuisine options active for similar reasons. She doe not know what type of restaurant will be considered the best after all.

Knowing that the map is now only showing the higher quality restaurants across all cuisine types in Toronto, she scrolls the map towards the different areas that she will be visiting. The first area she plans to visit is near Exhibition Place, so she zooms in the map to the region. The cuisine category chart updates accordingly in their distributions.

Sally can now see that Bars & Pubs seems to dominate the area which is rather unfortunate as she can get these places back home. But she does notice that a second larger portion of Breakfast places are also high in the area which could lead to promising results. African food also appears in this area, although it is nowhere as dominant. However due to the review quality filters being active, she trusts that some of these African places might also be interesting to check out.

With these 2 categories standing out, she selects to be the focus of investigation by clicking on them in the cuisine category chart, with the other cuisines deselected as they are no longer of interest. The map then updates with markers representing both of these cuisine types. She sees both a breakfast place and an African restaurant both nearby to each other, which might make it handy to check out both places in quick succession. She jots down the names of both places to visit them later. Please refer to Figure 11 for viewing Sally's decision making process.

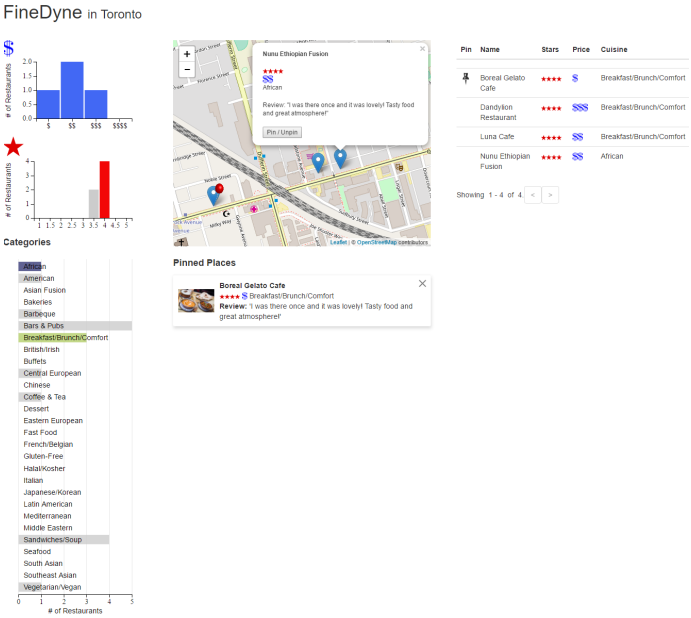

Fig. 11. Sally notices that African and Breakfast places are popular in her current region of interest. She selects them to filter and find matching restaurants to visit.

7 Discussion

As the above scenarios have demonstrated, FineDyne can be used to support restaurant discovery no matter the level of criteria details a user may have as they begin the dining decision process. In particular, FineDyne supports an iterative way to refine search criteria, where it is simple to change criteria constraints at any time. Or in cases where it is not clear where one should start their decision process, the charts shown in the criteria view give hints of promising criteria combinations by providing a holistic way of seeing how various distributions can change. All of this combines in a way that can greatly improve over the current Yelp website.

However, there are several limitations to FineDyne ranging from technical to design issues. The most notable design issue was the saliency of our final chosen aggregated cuisine categories. Aggregation had to be performed for scalability with our chosen idiom of barcharts. However, this maybe confusing for users. For example, if a user were to find out which cuisine category contained Vietnamese

food, would they look in Southeast Asian or South Asian? In other words, the transparency of a specific unique category may not align with our chosen aggregations. A solution to this may involve either nested cuisine categories (Figure 12) or simply having a preferences panel that defines the final number of unique cuisine types shown in the criteria view (Figure 13). Either solution would reveal the true location of a specific cuisine category.

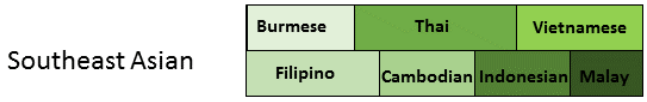

Fig. 12. Nested cuisine category example based on Southeast Asian cuisine. Size of proportions indicate the dominance of a particular cuisine within an aggregated cuisine category. Different luminance levels were used to distinguish the different cuisines inside the aggregation.

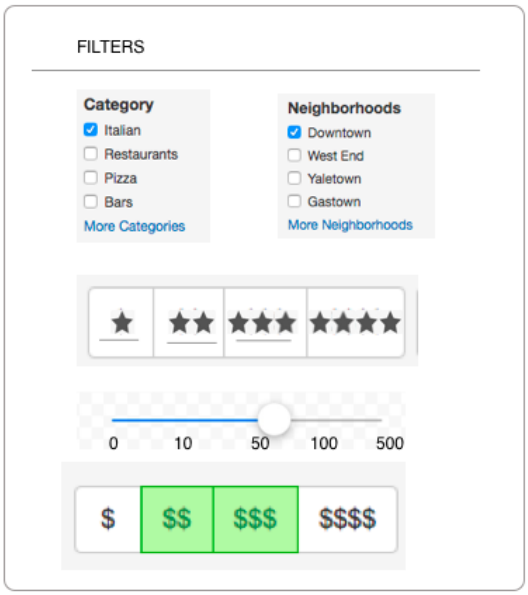

Fig. 13. Simple preferences panel. Based on old original design, where a slider was considered to filter restaurants based on total number of reviews as well.

Along similar lines, there were encoding limitations due to the large number of cuisine categories shown, even after our aggregation. Because colour hue was used as the main form of encoding, we simply did not have enough distinct hues nor luminance levels to visually distinguish all of the cuisine categories. Some cuisine groups with similar hues may be confused to be similar to one another as a result. Using the preferences panel from above may also be a way to get around this issue in order to avoid having hues repeat themselves.

There were also encoding issues in the map view of FineDyne where the default map marker icons (blue balloons) of Leaflet were used. We ran out of time to investigate how to modify the markers to represent other types of cuisine categories. This would have been useful to more quickly identify what types of restaurants the markers were representing. However, another encoding issue here would have emerged, where using different colour hues would not have been enough to distinguish the various cuisine types due to the issue of repeating colour hues mentioned. A solution to this would be to employ unique glyphs such as sushi or pizza icons to represent iconic imagery of cuisines such as Japanese and Italian respectively. However, coming up with recognizable glyphs that is understandable for a diverse audience with different cultural backgrounds would be a challenge onto itself.

In terms of usability issues, the number of pinned restaurants would have necessitated users to scroll down their web browser to see the later pinned options. This would not only be frustrating, but the advantage of the linked highlighting of the pinned restaurants would be rendered useless, as users could no longer see the map view of the highlighted markers. A solution might be to redesign the candidate view with pagination to select different pages of candidate restaurants. Along similar lines, the large number of cuisine categories also forced users to scroll down their webpages to see the various cuisine categories for selection. Here, rearranging the rowchart perspective into a horizontal barchart may have been more wise to minimize scrolling.

In terms of technical issues, the map view and its current zoom level and the way in which it controlled the summary list of restaurants on the right could not be adjusted. Thus, there were cases where the designers had difficulty in rendering the right list of restaurants even when a similar zoom level was used. This is a side effect of using a out of the box solution like Leaflet, when a customized option may have been more controllable.

For our data parsing, we performed a final duplication removal step on the aggregated dataset. This did have a side effect, in that some restaurants of various cuisine types maybe classified as the first primary cuisine type found in the dataset (first category in the array). While this may lose the original identity of some restaurants, we performed this last step for potential performance saving reasons. In the future, it should be investigated how we can accommodate a restaurant under multiple cuisine types.

8 Conclusion

FineDyne provides an effective way to discover a restaurant that satisfies different levels of criteria knowledge through an iterative and holistic approach. This is achieved through 3 different views that are linked together. The criteria view allows users to see a distribution of all of the possible restaurants in a city and their associated price, review quality, and cuisine types which can be filtered in accordance with a criteria. The distributions are also updated with a users other known criteria inputs such as their desired location to eat, or specific price ranges theyre willing to spend. The map view provides the main way to see the criteria matching restaurants which can also be used to make decisions regarding spatial distances. Hovering over the restaurant markers in this view shows pop-up information such as the restaurants name, photo, and 1-liner review that could be used as the basis for pinning, a method to save a potential restaurant of interest for later consideration. A summary list was also implemented in order to allow users to quickly identify restaurants of interest rather than having them manually hover over restaurant markers of which many may exist. The lists columns for the various restaurant information such as price and review quality could be sorted by ascending/descending order as well. In order for users to make a final dining decision, the compare view was made where pinned restaurants from the map view could be represented as summary cards. Users were free to pin as many restaurants as they needed, which at the end could be used to compare candidate restaurants from a variety of different possible dining criteria. The cards could be hovered over as well in order to see the linked restaurant back on the map view to identify candidate restaurants for proximity based decisions. Combining the usage of these views, is what made FineDyne an effective way to make the most informed dining decision possible.

In the future, we wish to take FineDyne further by incorporating other city datasets (Vancouver) as well as improve upon the limitations we have described. Suggestions from colleagues have also suggested using the live Yelp API such that the data reflected by FineDyne would be reflective of current real life data. It would also be interesting to consider how mobile devices would interact with the information found with FineDyne, as it is common that users tend to look for places to eat while on the go.

ACKNOWLEDGMENTS

The authors wish to thank Tamara Munzner for her helpful suggestions and feedback throughout the project process.

REFERENCES

[1] Crossfilter. http://square.github.io/crossfilter/. Online; accessed 28 April 2017.
[2] d3.js. https://d3js.org/. Online; accessed 28 April 2017.
[3] dc.js. https://dc-js.github.io/dc.js/. Online; accessed 28 April 2017.
[4] dc.leaflet. https://www.npmjs.com/package/dc.leaflet. Online; accessed 28 April 2017.
[5] leaflet. http://leafletjs.com/. Online; accessed 28 April 2017.
[6] R. https://www.r-project.org/about.html. Online; accessed 28 April 2017.
[7] Trifacta Wrangler. https://www.trifacta.com/products/wrangler/. Online; accessed 28 April 2017.
[8] Yelp. https://www.yelp.com/vancouver. Online; accessed 28 April 2017.
[9] Yelp Cuisine Categories. https://www.yelp.ca/developers/documentation/v2/category_list. Online; accessed 28 April 2017.
[10] Yelp Dataset Challenge. https://www.yelp.ca/dataset_challenge. Online; accessed 28 April 2017.
[11] Yelp Price Categories. https://www.quora.com/How-are-dollar-signs-calculated-on-Yelp-and-who-calculates-them. Online; accessed 28 April 2017.
[12] Yelp Ranking Information. https://www.yelp-support.com/article/How-are-search-results-ordered?l=en_US. Online; accessed 28 April 2017.
[13] M. Drk, S. Carpendale, C. Collins, and C. Williamson. Visgets: Coordinated visualizations for web-based information exploration and discovery. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1205–1212, Nov 2008.
[14] M. A. Hearst. Design recommendations for hierarchical faceted search interfaces. Master's thesis, UC Berkeley School of Information, 2006.
[15] T. Munzner. *Visualization Analysis and Design*. CRC Press, 2014.
[16] T. Ortner, J. Sorger, H. Steinlechner, G. Hesina, H. Piringer, and E. Grller. Vis-a-ware: Integrating spatial and non-spatial visualization for visibility-aware urban planning. *IEEE Transactions on Visualization and Computer Graphics*, 23(2):1139–1151, Feb 2017.
[17] B. Saket, H. Kim, E. T. Brown, and A. Endert. Visualization by demonstration: An interaction paradigm for visual data exploration. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):331–340, Jan 2017.
[18] B. Shneiderman. Dynamic queries for visual information seeking. *IEEE Software*, 11(6):70–77, Nov 1994.
[19] W. Willett, J. Heer, and M. Agrawala. Scented widgets: Improving navigation cues with embedded visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1129–1136, Nov 2007.