Predicting Success of Restaurants on Yelp using Attribute-Specific Spatial Clusters

Heidi Chen (hchen7), Edward Lee (edlee1), Tyler Yep (tyep)[1]

***Abstract*— Many studies have sought to predict the success of a restaurant given its surrounding competition and characteristics [1] [2]. However, such studies have not taken into account the topologies in restaurant distribution, often assuming that only basic heuristics like surrounding restaurants within a given radius affect the performance of the restaurant. Restaurants may not always be distributed as such, and restaurants outside of such thresholds may still affect the performance of the given restaurant. As an example, restaurants far from each other on a busy street may affect each other more than restaurants nearby but not on that street. Not just that, such studies often do not distinguish between type-specific restaurant density and global restaurant density. For instance, Chinese restaurants may be dense in an area because they are in a community like Chinatown, or simply because they are in a very dense area with many restaurants. The goal of our project is to account for irregular restaurant distribution in determining how being in a community of similar restaurants as compared to a more diverse array of restaurants affects a restaurant's success. We intend to achieve this by: (1) identifying communities through a combination of spatial clustering and multiplex graph community detection techniques, (2) evaluating the theoretical success of these community detection algorithms on a set of null models, and (3) comparing predictions of success among restaurants between, inside, and outside such communities.**

I. Introduction

A key problem in marketing is that of *geographic market segmentation*. The goal of geographic segmentation is to divide a region according to social and demographic characteristics. This activity is especially important in the restaurant industry, where the geographic placement of a restaurant can significantly affect its future success. One factor that contributes to the importance of geographic placement is the placement of similar restaurants in the surrounding area, which contributes to the competition a restaurant faces [2]. For example, two Chinese restaurants close together may compete for the same market of customers looking to eat Chinese food. However, it may also be the case that a cluster of Chinese restaurants may draw more overall customers looking to eat Chinese food to the area and thus increase business to both. We thus want to quantify the effect surrounding, similarly-typed restaurants can have on an individual restaurant's success.

Past works [1] [2] have approached the definition of surroundings in a very simple manner, using the distribution of restaurants within some given radius as their feature. While this allows the metric to take into account how the density of similarly-typed restaurants compares with the overall density of restaurants in the area, it still fails to take into account two factors. The first is how competition is likely not directly correlated with distance. For example, a pair of restaurants far from each other on a busy street are likely bigger competitors of each other than a third restaurant nearby that is not on that street. The second is that of restaurant density. The radius threshold in a metropolitan area may be significantly smaller than in a suburban or rural area, meaning a single threshold is not applicable to all settings. Thus, we plan on using and developing algorithms that can be used to connect and cluster similarly-typed restaurants together normalized to overall restaurant density, as well as connect and compare the success of restaurants inside and outside these clusters. As far as we can tell, not much work has been done in the space of identifying type-specific clusters. Thus, we plan to draw upon existing research done in spatial clustering and community detection.

In Section II, we provide an overview of previous work that has inspired this paper, grouped into three categories: studies on predicting restaurant success, work done on graph-based spatial clustering methods, and community detection algorithms that could be applied to graphs with attributes. In Section III, we describe our approach to this problem, including the dataset, our evaluation methods, and an in-depth explanation of the algorithms we have run. In Section IV, we discuss the results of our graph construction, community detection, and the accuracy of our predictions of restaurant success based on these algorithms and various supervised learning models. In Section V, we identify key takeaways from our research and explore possible future directions for this topic space.

*This work was not supported by any organization

[1]Heidi, Edward, and Tyler are with the Department of Computer Science, Stanford University

II. Related Work

Since we are focused on applying spatial clustering and community detection algorithms to restaurants, we decided to look at three different areas: (1) existing approaches to relating restaurant surroundings to restaurant ratings, (2) graph-based spatial clustering techniques, and (3) community detection in type-specific settings.

A. Restaurant Surroundings

There were 2 main restaurant-related papers that affected our problem setup.

Wang et. al [1] used features like review-based market attractiveness, review-based market competitiveness and quality, and density to run regressions to output the predicted number of Yelp check-ins per month for a given candidate location (specified as an area with a 200m radius). Their results indicated that textual, review-based features using sentiment analysis were better predictors of Yelp check-ins than the geographic features. However, they run into the aforementioned problem of using a simple metric like radius that may not accurately capture how surrounding geography affects check-ins. Additionally, this paper's results are questionable because the authors trained their model on exclusively non-chain restaurants, and then tested and evaluated it on chain restaurants which may not have been totally valid. Even so, the paper provides good ways of testing the effect of clusters on restaurant success. We therefore hope to re-investigate their claims with our dataset, using clustering techniques to more accurately capture geographic competitiveness and taking care to avoid the same assumptions in our evaluation.

Athey et. al [2] leveraged GPS movement data in order to predict demand for certain types of restaurants during lunchtime. The study determined that restaurant choices available and people's personal preferences are the two major factors that lead to choosing a restaurant for lunch (personal preferences include the type of food, price ranges, travel times, etc). The paper conveys two major findings. First, a person's willingness to travel differs significantly depending on the type of restaurant, as Japanese, New American, and Vietnamese restaurants have a much larger user travel radius compared to sandwich, pizza, or Mexican cuisines. This is an excellent basis for our project, as this finding means that it is likely communities of similar-cuisine restaurants in close proximity will have more success because of their similar travel times. It also indicates that we may want to investigate travel time as a distance metric rather than geographic distance.

B. Spatial Clustering

Many spatial clustering methods have been developed over the years to group points in some spatial domain into clusters. There are many different types of spatial clustering methods, ranging from partitioning-based to grid-based algorithms. Of particular note are those that are graph-based, which allow for an adaptivity to various applications that other methods often lack. For example, partitioning-based algorithms like k-means typically require the user to select a k, and grid-based algorithms require the user to select some granularity for the grid size.

An early graph-based approach by Harel and Koren [3] attempts to use short random walks to modify existing edge weights such that edges between nodes in a cluster will be higher than edges between different clusters. They use two different metrics: Neighborhood Similarity, which is essentially a proximity-based distance metric, and Circular Escape, which relies on the intuition that a path between nodes of different clusters will be less likely that paths between nodes of the same cluster. Neighborhood Similarity seems to be faster but perform worse than Circular Escape. One flaw of the algorithm is that it due to the short path-lengths and the need to run random walks on different subgraphs, long-range connections in larger clusters are much more difficult to capture.

Nowadays, cutting-edge graph-based spatial clustering algorithms seem to focus on extracting explicit features from a constructed graph and clustering based on those features. The constructed graphs tend towards 2 main algorithms: k-Nearest Neighbors [4], and, more recently, Delaunay Triangulation [5] [6]. k-Nearest Neighbors-based algorithms like CHAMELEON [4] combine the algorithm with a subsequent agglomerative clustering on the properties of the generated graph to cluster points. While this is effective, it suffers the same problem as the aforementioned partitioning- and grid-based techniques, which require prior knowledge. On the other hand, Delaunay-based algorithms like AUTOCLUST [6] exploit information like the variability of edge weights to identify boundaries in clusters, which allows the algorithms to adapt to different datasets without requiring user input.

Although all the algorithms mentioned are likely not directly applicable to our problem, as they typically focus on non-attributed nodes, they still provide valuable insight into how to construct graphs and identify possible attributes that can be used to detect attribute-specific communities.

C. Community Detection

Many complex networks cannot be represented using a single edge type. Instead, to represent the complex interactions between nodes, networks with multiple layers each sharing the same set of nodes but different edge sets have risen in popularity. These types of networks are known as multiplex networks. Although we were not able to find graph-based spatial clustering algorithms that map directly to the multiplex space, there has been work on identifying layer-specific communities.

Kuncheva and Montana [7] do this by extending existing monoplex random walk algorithms by allowing the random walker to travel across layers, using transition probabilities dependent on local topological characteristics, thus the name Locally Adaptive Random Transitions (LART). Probabilities are calculated to encourage jumps between nodes with similar topologies and discourage the opposite. This allows the random walker both explore multilayer-spanning communities and get "trapped" in single-layer communities. The random walk generates a distance measure that can then be used to cluster nodes in the network together through typical clustering methods. Kuncheva and Montana find that the algorithm is "better able to detect layer specific communities and communities that are shared across several but not all layers", as desired. This makes LART quite relevant to our problem space, where restaurant communities likely do not share all the same characteristics, and thus finding layer-specific communities is more relevant.

However, it is difficult to say how well random walks, or more generally, any community-detection algorithm would work on spatial data. Due to the graph construction techniques used on spatial data, nodes only have edges to other nodes in their physical proximity, meaning the interconnectedness of target clusters are likely not very different from those of its surroundings. And since modularity metrics, which community-detection algorithms use for measuring the "goodness" of a split, depends on interconnectedness of target communities being higher than usual (the exact opposite), we don't know how well community detection algorithms will work in our problem. Even so, Kuncheva and Montana provide valuable insight into the topological characteristics across multiple layers of multiplex networks.

III. Approach

A. Datasets

We used the Yelp Dataset [8] of 188,593 businesses in 10 metropolitan areas. However, since we require distances between restaurants, for our exploration of algorithms we focus on a single area, namely Toronto, Canada, due to its sizeable number of restaurant nodes (7578) and the existence of a Chinatown in the area, which gives a quick and easy qualitative measure of the performance of our community detection algorithms. In our results section, we will also explore Calgary and Montreal, other Canadian cities of similar demographic and data set size (2793 and 3682 restaurant nodes, respectively.)

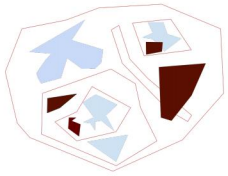
B. Evaluation Methods

In order to quantitatively evaluate our clustering algorithms, we want to evaluate both how well each algorithm performs when identifying attribute-specific communities, as well as predicting restaurant success.

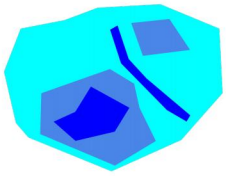
Data: 2 maps of pixel to category M_C and to density M_D as in subfigures (a) and (b) of figure 1
Result: Map from generated points to categories M_P
Extract C from images. Choose D, p_b, and p_c;
$M_P = \{\}$;
foreach *pixel* $p(x, y)$ **do**
  cat = $M_C[p]$;
  dens = $M_D[p]$;
  if *rand(0,1)* $< p_b$ **then**
    Initialize empty list for $M_P[p]$;
    foreach *category* $c \in C$ **do**
      if *rand(0,1)* $< D_{dens} + [cat = category]p_c$ **then**
        Add c to $M_P[p]$
      end
    end
  end
end
return M_P;

Algorithm 1: Point generation algorithm

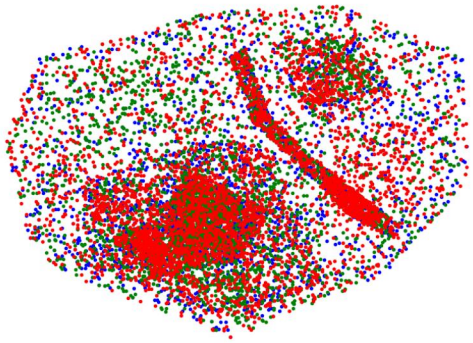
To evaluate how well our clustering algorithms identify attribute-specific communities, we need to be able to compare our detected communities to some ground-truth. Although Yelp does have a community attribute for each node, this community is not the same as the community we are looking for, as we want attribute-specific communities rather than demographic communities specifically. To get this, we must have spatial data with ground truth communities. As far as we know, no such data is available. Spatial clustering papers like [4] and [6] do have point distributions with varying densities, but none offer points with varying attributes. Thus, we create our own graphs with the following parameters: categories set C, densities D where D_i is the probability each pixel is a point for density level i, pixel-to-category distribution M_C, pixel-to-density distribution M_D, base



(a) category distribution (light blue and brown are categories)



(b) density distribution (darker is more dense)



(c) points generated (red = brown, green = light blue, blue = noise)

Fig. 1: Point generation

probability of a point being a given category p_b, and additional probability of a point being a given category in a cluster of the same category p_c as in algorithm 1. An example of this process can be seen in figure 1. Once we have our clusters, we use metrics F1 score to see if the clustered areas are correctly labelled for each category, including no category.

We also want to identify the impact our detected communities have on the success of a restaurant. Our plan follows the approach used by Wang et al. [1] by running a regression on the data using graph-based features like clustering coefficient and cluster size to determine the weight and sign each feature has on predicting success metrics like rating, review count, and number of check-ins. We will focus on review counts for our regressions. By doing our regression on multiple different community detection approaches, we identify how well each algorithm does in encoding information that can be used to predict restaurant rating.

C. Algorithms, Techniques, and Models

Due to there not being much directly related work in this area, we decided it would be better to have many different algorithms and evaluate how well each does at both identifying correct nodes in the generated graphs and predicting restaurant ratings on actual data.

Also, to fit the limited time and resources we have, we decided to focus solely on layer-specific communities rather than attempting to identify communities that may occur across multiple layers of the graph. This allows us to simplify the problem down from a graph with arbitrary layers and instead focus solely on the base layer (that with all nodes) and the layer relevant to the attribute we are examining. In the Yelp dataset for example, if we were looking for Chinese-specific communities, this would be the layer with all restaurants and the layer with solely Chinese restaurants.

1) Algorithm: k-Nearest Neighbors with Filter: It is possible to identify attribute-specific communities using k-nearest neighbors by first running k-nearest neighbors on the full graph ignoring attributes. Intuitively, this means restaurants in an attribute-specific cluster will have a higher chance of connecting to restaurants with the same attribute, while restaurants in global clusters (i.e. dense areas with restaurants of all different attributes) will be more likely to connect to restaurants of different attributes. Then, to find the layer-specific communities for a given attribute, we would be able to choose only those edges connecting nodes of that given attribute and, with a good k value, identify the clusters as the connected components of the resulting graph. Then, by filtering out those connected components smaller than a given threshold (in this case, the sum of the average and standard deviation of the sizes of all connected components), we should be able to identify the most prominent communities.

2) Technique: Delaunay Triangulation: The other way of constructing graphs to capture distance information between nodes is through Delaunay Triangulation, which has risen to popularity in spatial clustering with techniques like AUTOCLUST [6] and ASCDT [5] using it to cluster points with great success. Although Delaunay Triangulation has performed quite well in attribute-less spatial clustering, its utility in attribute-specific community detection remains to be seen. We cannot rely on splitting a globally generated Delaunay diagram based on categories like in k-nearest neighbors as there are much fewer edges between each node.

3) Technique: Edge- and Angle-based Normalization: Instead, to take advantage of the rich distance information captured by Delaunay Triangulation, we attempt to apply effective single-layer spatial clustering methods to our graph. To do this, we must first combine the 2 relevant graphs, the base layer and the attribute-specific layer. We decided the best way to do this was by normalizing the edge weights of the attribute-specific layer on the edge weights of the base layer. The intuition behind this is that the length of an edge corresponds to the density of the graph at that given region. Namely, as

the density of nodes increase, the average edge length of the graph constructed should decrease. Thus, by normalizing, we are able to cancel out the change in density due to overall density shifts and only examine the changes in attribute-specific density.

We explore 2 possible ways of doing this: normalization based on the average of all edges of a given node in the base layer, which we will denote edge-based normalization, and normalization based on only the lengths of the edges surrounding the given edge of a given node in the base layer, which we will denote angle-based normalization. These normalized graphs will be used in the following algorithms.

More formally, given a layer l (e.g. the base layer), we can use Delaunay Triangulation to construct a graph $G^{(l)}$ with vertices $V^{(l)}$ and edges $E^{(l)}$ where edge $(u, v) \in E^{(l)}$ has weight $W^{(l)}(u, v)$. Let $mel(l, u)$ be average length of an edge from node u in $G^{(l)}$. Also let the base layer be b and attribute-specific layer be a. We know that $V^{(a)} \subseteq V^{(b)}$.

We can define edge normalization based on these definitions. Namely, given base layer b and attribute-specific layer a, we can construct the edge-normalized graph $G^{(ce)}$ (which we want to be unweighted) as: $V^{(ce)} = V^{(a)}$, and that for each $(u, v) \in E^{(a)}$:

$$W^{(ce)}(u, v) = \frac{W^{(a)}(u, v)}{mel(b, u)} + \frac{W^{(a)}(v, u)}{mel(b, v)}$$

Angle normalization is conceived as a way to capture more directional information than edge normalization. By only focusing on a select few edges in the same direction, we can better capture the densities of the graph in different directions of a given node. We expect this to help angle normalization-based methods to be more accurate than edge normalization techniques when distinguishing nodes at the boundaries of clusters.

Angle normalization requires choosing the 2 or 3 closest edges of the base layer in terms of angle. Given $u, v \in V$, let $angle(u, v)$ be the angle of the edge from node u to v. We can find this given the coordinates of u and v (x_u, y_u) and (x_v, y_v) in code as `math.atan2(`$y_v - y_u$`,`$x_v - x_u$`)`. Thus, the algorithm for choosing the closest edges of the base layer to a given edge can be expressed as in algorithm 2. Given set $S_{(u,v)}$ and $S_{(v,u)}$ and $ave(S)$ being the average lengths of all edges in S, we can define angle-normalized graph $G^{(ca)}$ as: $V^{(ca)} = V^{(a)}$, and that for each $(u, v) \in E^{(a)}$,

$$W^{(ca)}(u, v) = \frac{W^{(a)}(u, v)}{ave(S_{(u,v)})} + \frac{W^{(a)}(v, u)}{ave(S_{(v,u)})}$$

Data: $E^{(b)}$, $E^{(a)}$, $(u, v) \in E^{(a)}$
Result: Set of edges $S_{(u,v)}$ whose angles are similar to $angle(u, v)$
Extract C from images. Choose D, p_b, and p_c;
$A = []$;
foreach $\{(m, n) | (m, n) \in E^{(b)}, m = u\}$ **do**
  Add $(angle(m, n), (m, n))$ to A;
end
sort A on first element;
if $\exists g \in A$ *where* $g.first = angle(u, v)$ **then**
  Get elements $f, h \in A$ 1 place before and after g in A;
  return $\{f.second, g.second, h.second\}$;
else
  Get the 2 elements $f, g \in A$ surrounding where $angle(u, v)$ would be if placed in A;
  return $\{f.second, g.second\}$;
end
Algorithm 2: Algorithm for choosing edges to use in angle normalization

4) Algorithm: Edge Removal with Filter: The simplest algorithm we can do with the normalized graphs is to remove all long edges. The intuition here is that longer edges in the normalized graph correspond to edges in low-density regions. And since we are more focused on finding high-density communities, we can safely ignore these edges. Thus, given the normalized graph, we can continuously remove edges which are longer than expected until connected components form and consider these connected components as our clusters. We find through our experiments that removing all edges above the average edge length in two rounds before filtering out all smaller-than-average connected components gives the best results, though this varies from graph to graph.

If, however, we wanted to find both high- and low-density clusters, we would likely need a different approach similar to that from AUTOCLUST, where rather than examine length of edges, we examine how they deviate from the norm at a given node, as edges from a node connecting different clusters are likely to have different normalized lengths than those at the same node connecting to within the cluster.

5) Algorithm: Random Walk: To explore the possibilities of detecting attribute-specific communities through random walks, we begin with previously mentioned algorithms described by Harel and Koren [3] based on Neighborhood Similarity and Circular Escape. Once we create a graph G' using the normalized graph G where $W'(u, v) = \frac{1}{W(u,v)+1}$ to promote the likelihood of the random walker going towards closer nodes than farther nodes, we can run the random walk based either on

Neighborhood Similarity and Circular Escape.

In Neighborhood Similarity, we run random walks with a length of 3 to construct vectors for each node based on the probability of reaching other nodes in the graph and use the similarity of these vectors as the new weights of the vectors. We can use such small walk-lengths because we only care about similarities between nodes that share an edge. We can then cut off all weights below a certain threshold to get a good clustering.

On the other hand, with Circular Escape, we run random walks on the subgraph surrounding each edge and identify the probability of reaching the target node and then returning back to the start node. We expect this probability to be lower when the edge is between nodes in different clusters and high when the edge is between nodes of the same cluster.

The other approach we considered exploring was modifying LART [7] to promote exploring as far as possible in the base layer and promote staying within a cluster in the attribute-specific layer. However, after observing the sub-optimal performance of both Neighborhood Similarity and Circular Escape in spatial data, we decided that, in the interest of time, we would skip LART in favor of non-random-walk-based algorithms.

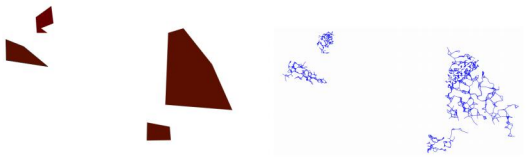
6) Algorithm: Louvain: We also attempted to explore how modularity optimization would perform on both graphs constructed from k-nearest neighbors as well as the aforementioned normalized graphs, so we opted to use the Louvain method for community detection [9] to construct communities out of these graphs. We don't expect modularity to perform well as a metric of clustering success since the graphs likely don't have much variation in edge density between nodes of high density and nodes of low density.

IV. Results

A. Predicting High-Density Clusters on Constructed Spatial Data

To evaluate the performance of our algorithms, we conducted experiments using graphs we generated ourselves. We constructed 7 different point distributions using the aforementioned point generation algorithm. A point is considered in a high-density cluster if it was generated in a colored region of the category map and is of that category. For example, a light-blue point would be considered high-density if it was generated in a light-blue region of the space.

We then ran our algorithms on the point distribution and identify these high-density clusters and return the predicted set of high-density points. An example of



(a) flipped brown category distribution from figure 1

(b) clusters identified by k-nearest neighbors with $k = 8$

Fig. 2: Comparison between category distribution when generating points and final clusters identified

what was generated is given in figure 2. Each algorithm was given a list of hyperparameters to try. In k-nearest neighbors, all k from 4 to 15 are explored, with $k = 9$ being the optimal most often. In edge removal of both edge- and angle-normalized graphs, 1 to 3 rounds of removing edges above average were done, with 2 rounds usually being most optimal. In Neighborhood Similarity, cutoffs from 0.1 to 0.7 were explored, with a cutoff of around 0.58 being most optimal. On the other hand, in Circular Escape, 0.21 was usually the best threshold. And last, the Louvain method on the k-nearest neighbors graph had an optimal k that varied wildly from 8 to 19 when exploring ks from 5 to 20. The top F1 score of each algorithm is listed.

We then use these F1 scores to compare to the ground truth. The results are described in figure 3, where results are ordered by F1 score in graph "sectioned". Baseline is also included, where baseline essentially considers all points of that given category high-density.

	complex1	complex2	boundary1	boundary2	boundary3	sectioned	simple
er,angle	**0.8949**	**0.9046**	**0.9865**	**0.9151**	**0.9542**	**0.9638**	**0.9858**
knn	**0.9389**	**0.9338**	**0.9746**	**0.8988**	**0.9472**	**0.9605**	**0.9893**
er,edge	**0.9120**	**0.8877**	**0.9752**	**0.8346**	0.9053	**0.9018**	**0.9673**
ns,edge	0.4649	0.6019	0.7292	0.7996	0.9017	0.8539	0.8271
ns,angle	0.5569	0.6894	0.8368	0.8124	**0.9061**	0.8154	0.9095
ce,angle	0.5790	0.5801	0.7801	0.8003	0.8974	0.7964	0.7944
ce,edge	0.4881	0.5045	0.7591	0.7987	0.8974	0.7317	0.7841
nl,edge	0.4353	0.4953	0.6836	0.6236	0.8024	0.7057	0.8263
baseline	0.3873	0.4128	0.6714	0.7982	0.8974	0.7040	0.7632
nl,angle	0.4310	0.5137	0.7247	0.6723	0.7707	0.7027	0.8599
kl	0.4504	0.6159	0.4789	0.4696	0.3446	0.4535	0.6323

Fig. 3: F1 scores of the different algorithms on identifying nodes in high-density clusters for a single category in 7 different generated graphs (with hyperparameter search). Top 3 performing algorithms for each graph are bolded.

Based on the figure, we see that k-nearest neighbors and edge removal with both edge and angle normalization are consistently the top 3 performing algorithms. k-nearest neighbors seems to perform the best most often, then angle-normalized edge removal, and last edge-normalized edge removal. We see that angle-normalized edge removal usually performs better than edge-normalized edge removal, indicating that angle

normalization is able to capture important directional information that allows it to make better cuts on spatial data than edge normalization.

These three being the most effective is somewhat expected due to the number of points in each graph, which is consistently above 10,000. The size of the graphs generated mean that the small walk-length in the random walk-based approaches likely cannot capture the many enormous high-density regions in each distribution. Similarly, the size and spatial nature of the graphs generated may mean that the Louvain method for community detection is either too inexact or just plain unfit for the task at hand.

The top 3 performing algorithms are then the algorithms we focused on using to predict restaurant success between restaurants inside and outside of attribute-specific communities and. Special attention is also given to the Louvain method since it provides not just communities of high-density, but low-density as well, which may be useful when analyzing the properties of non-community restaurants.

B. Qualitative Evaluation of Algorithms on Yelp Data

To qualitatively evaluate our graph construction and edge selection, we focused on identifying Chinese-specific communities due to the easily recognizable Chinese restaurant district in Toronto: Chinatown. While this spot is quite small relative to all of Toronto and thus may be difficult when zoomed out on the map, its location in the urban area of Toronto means that it will be a good way of testing whether our algorithms are identifying Chinese-specific communities or just high-density areas in general. We expect our algorithms to find Chinatown and other Chinese-specific communities just like it, including Richmond Hill, a community northwest of the Toronto urban area with a well-established Chinese community.

We begin with the highest-performing algorithm: k-nearest neighbors with $k = 9$, whose results can be seen in figure 4. The two main connected components (lower left and top right) are the locations of Chinatown and Richmond Hill. Thus, we see that k nearest neighbors is able to identify these attribute-specific communities quite well, even if they might be sparser than otherwise expected.

We also attempt to identify communities using edge removal, as can be seen in the four graphs of figure 5. We see that while we are able to somewhat identify Chinatown and Richmond Hill, 2 rounds of filtering is too little and 3 rounds is too much. Not just that, we are unable to isolate just Chinatown itself, and instead pick up

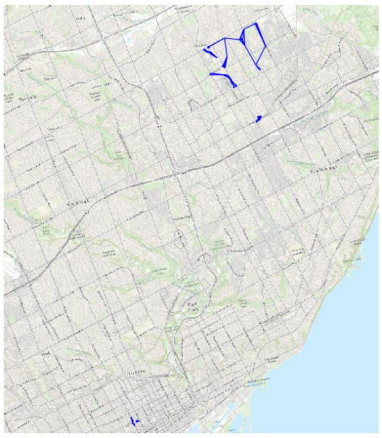
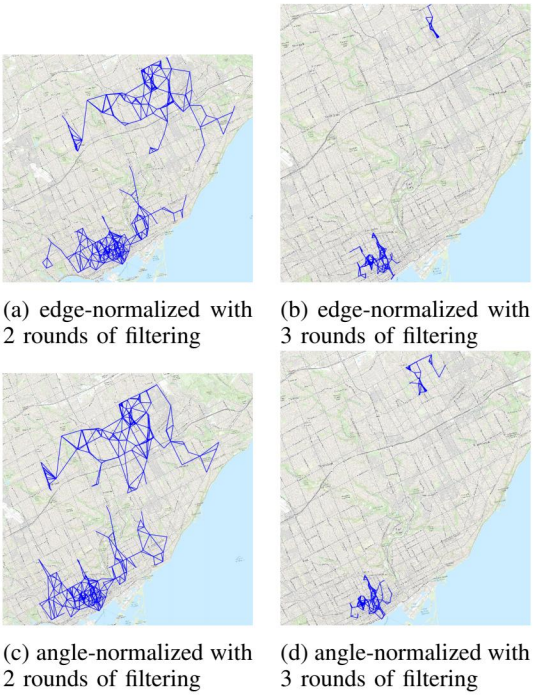


Fig. 4: k-nearest neighbors with $k = 9$



(a) edge-normalized with 2 rounds of filtering

(b) edge-normalized with 3 rounds of filtering

(c) angle-normalized with 2 rounds of filtering

(d) angle-normalized with 3 rounds of filtering

Fig. 5: Edge removal on different normalizations and number of rounds of filtering

restaurants in the surrounding urban area. While it may be that Chinese restaurants are denser here, we believe it's more likely that the algorithm has factored in overall density into the equation. Thus, edge removal performs worse than expected, indicating that we may need to make a more granular measure of filtering and/or adjust the normalization factor, perhaps by square-rooting each component in the normalization formulas to adjust for how density correlates with average edge length in Delaunay graphs.

C. Predicting Success of Restaurants with Community-based Features

Having evaluated the accuracy of various graph construction and community detection algorithms, we selected the most successful methods (k-Nearest Neighbors and edge removal with normalized angles and edges) to establish a prediction model for restaurant success.

We chose three similar Canadian cities, Toronto, Calgary, and Montreal, for our train, dev, and test sets respectively. To account for type-specific restaurant density vs. global restaurant density, we considered two versions of this data set: one where we included all restaurant nodes in these cities, and one where we filtered the cities' data down to restaurants that fell in the top 10 most common Yelp categories (i.e., Chinese, Mexican, Coffee & Tea, etc.). Before filtering, Toronto had 7578 nodes, Calgary had 2793 nodes, and Montreal had 3682 nodes. After filtering, Toronto had 3720, Calgary had 1220, and Montreal had 1393 nodes, for a total of 5113 restaurants in our filtered data set.

The features we considered were:

1) Node Degree
2) Node Clustering Coefficient
3) Community Edge Density
4) Average Review Count in Community
5) Community Size

For the full dataset, we extracted features on communities built from the entire graph. For the filtered dataset, we split each graph into a set of induced subgraphs based on their category and extracted features per node from communities in the subgraphs. We then concatenated the separate categories' node-level features into a unified feature matrix for each graph.

We defined our labels for restaurant success as review count, which approximates the number of customers a business receives. To focus on the effect of community features, we decided not to add star rating or number of check-ins to our labels, as review counts seemed to theoretically be the most likely indicator of success.

For each graph construction and community detection method, we evaluated several machine learning models. We found that Linear Regression performed the best overall in classifying restaurant success using community features. Decision Trees were also notably adept, but they tended to overfit to the training set (with MSEs of almost 0) and yielded much higher test errors (in the tens of thousands).

We also experimented with more complex models, such as SVMs, Logistic Regression, and Adaboost, but they perform roughly the same as Linear Regression. More hyperparameter tuning may be necessary to unlock the full potential of these models.

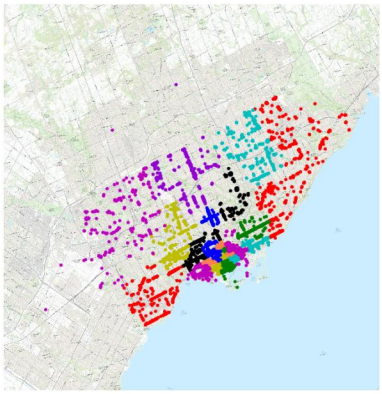


Fig. 6: Louvain communities on KNN-constructed graph

1) k-Nearest Neighbors: We selected $k = 9$ for our algorithm because of its high performance in our theoretical evaluation, and ran KNN to construct graphs from both the full data set and the category-filtered data. We considered any connected component to be a community, and performed feature extraction based on these community assignments. We ran this algorithm both keeping all connected components, and filtering out the smaller components (as described in our approach section). We found that KNN run on the full graph, without filtering out smaller connected components, was among the better predictors of restaurant review counts.The discrepancy between this result and the filtering success in our theoretical evaluation may be due to the smaller size of the Toronto data set.

2) Edge/Angle-Normalized Edge Removal: Starting from the Delaunay graphs, and normalizing edges using the methods mentioned above, we identified the remaining communities based on edge density and attempted to predict restaurant review counts. In terms of MSE, these community features did not help any of our predictors classify review counts. In all three of the cities, adding the community features ultimately resulted in a worse MSE than KNN or Louvain.

3) Louvain: After using the Louvain Algorithm to partition our KNN-graph into communities, we extracted community-level features and surprisingly found that the Louvain algorithm yielded the highest net result on all three of our selected cities. In Fig. 7, we show the results of Linear Regression on each of our community detection algorithms.

As our baseline, we ran a linear regression on a feature matrix of one feature that was always set to zero. We

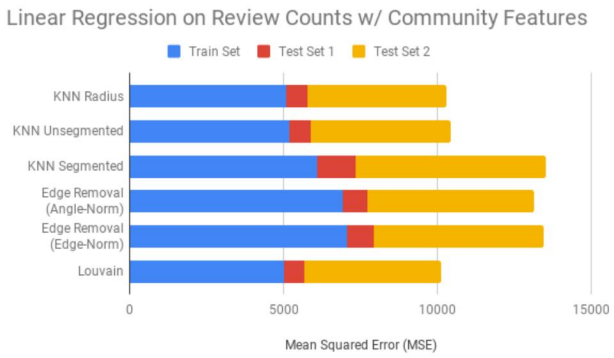


Fig. 7: Linear Regression results.

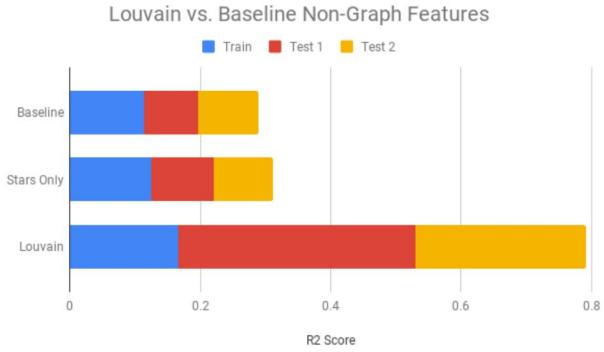


Fig. 8: Comparison with baseline.

also ran a linear regression with a feature matrix with only star ratings. The comparison of these regressions with Louvain in terms of cumulative R^2 score is shown in Figure 8; Louvain does significantly better than both non-graph-based approaches.

The weights from Louvain (Figure 9) indicate that clustering coefficient on an uncategorized graph may contribute the most to an accurate prediction of restaurant success. One possible interpretation of this is that overall density is a better measure for restaurant success than anything category-specific.

Feature	Degree	Clustering	Comm. Edge Density	Comm. Size	Comm. Review Count
Weights	-0.2569	6.704	0.8015	0.0256	0.1419

Fig. 9: Weights from Louvain linear regression.

V. Conclusion

Through a detailed process of graph construction, community detection, and prediction algorithms, we have experimented with and explored several different possible algorithms for identifying attribute-specific communities in spatial data, as well as several key observations about the role of such communities on restaurant success.

A. Algorithms

Through experimentation with different ways of constructing and normalizing graphs, as well as identifying edges and nodes to remove, we have identified the top 3 performing algorithms on our generated point distribution: k-nearest neighbors with a filter, and basic edge removal on an edge- or angle-normalized graph. These algorithms are able to consistently achieve an F1 score of above 90% compared to baselines that can range from 30% to 89% indicating that these methods is not just successful, but also consistent.

However, there are still issues with our algorithms. The most prominent indicator of this is that edge removal is unable to accurately capture Chinatown from amidst the Toronto dataset. Due to our lack of time, we were unable to generate points that accurately reflected the spatial topology of the restaurant distribution, which is a lot more geometric than the randomly-generated point distributions we used to evaluate our algorithms. Another problem may have been the large scale of the point distributions used. With such large point distributions, identifying exact boundaries of each cluster becomes less important than finding the general area of a cluster. With smaller graphs, the higher number of points on boundaries would likely increase the importance of accurately identifying edges between clusters, allowing for better identification of clusters in smaller graphs like the restaurant dataset. Additionally, we were unable to address many of the edge cases that modern single-layer clustering methods have placed much importance on, like necks and bridges. Since most of our algorithms are based on normalizing one layer over another layer, we could not extend most of these algorithms to identify spatial clusterings in more than 2 layers.

Despite all these problems, we believe that the work we have done is a good first step in exploring the space of identifying attribute-specific clusters in spatial data, and that this work can be extended to be more robust and work with more than one base layer and attribute layer.

B. Restaurant Success

We found that the Louvain-detected communities on the full, unsegmented graph yielded significant improvement to the MSEs of our restaurant review count success metric. Although Louvain performed poorly on the large graphs of our theoretical evaluation, it did well on our smaller graphs based on city-sized Yelp datasets (less than 10,000 nodes), and predicted review counts significantly better than non-graph-based features or the baseline. This indicates that Louvain shows promise for identifying communities that can be used to predict

restaurant success, as opposed to any attribute-specific communities.

Although we initially expected category-segmented KNN to yield the best results in prediction, it was outperformed by the unsegmented KNN and Louvain algorithms. Possible explanations for this discrepancy include that there may be factors contributing to restaurant success that are not solely linked to restaurant categories. Restaurants may do well because they are near a diversity of restaurants, or because they are simply in a dense, popular food hub; this information is not captured as well by our segmented subgraph features. In the future, we would like to implement more robust and complex multiplex graph construction algorithms like LART [7], for identifying more nuanced type-specific restaurant communities which may improve our prediction.

For additional future studies, we may explore different distance or travel metrics and repeat our graph community techniques on American and larger cities to see the optimal graph sizes for community effects.

REFERENCES

[1] Feng Wang, Li Chen, and Weike Pan. Where to place your next restaurant?: Optimal restaurant placement via leveraging user-generated reviews. *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 2371–2376, 2016.

[2] Susan Athey, David Blei, Robert Donnelly, Francisco Ruiz, and Tobias Schmidt. Estimating heterogeneous consumer preferences for restaurants and travel time using mobile location data. *AEA Papers and Proceedings*, 108:64–67, 2018.

[3] David Harel and Yehuda Koren. Clustering spatial data using random walks. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, pages 281–286, New York, NY, USA, 2001. ACM.

[4] George Karypis and Eui-Hong. Chameleon : A hierarchical clustering algorithm using dynamic modeling. 1999.

[5] Min Deng, Qiliang Liu, Tao Cheng, and Yan Shi. An adaptive spatial clustering algorithm based on delaunay triangulation. *Computers, Environment and Urban Systems*, 35(4):320 – 332, 2011.

[6] V Estivill-Castro and I Lee. Argument free clustering for large spatial point-data sets via boundary extraction from delaunay diagram. *Computers, Environment and Urban Systems*, 26(4):315 – 334, 2002.

[7] Zhana Kuncheva and Giovanni Montana. Community detection in multiplex networks using locally adaptive random walks. *CoRR*, abs/1507.01890, 2015.

[8] Yelp. Yelp open dataset. 2018.

[9] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.