An Analysis of Hierarchical Clustering Algorithms for Hotspot Detection in Geographical Request Maps

Paulos Kidane & August Bonds

FOR

Bachelor Thesis KTH Spring 2015
Abstract
The purpose of this thesis was to investigate the suitability of using different hierarchical clustering algorithms for locating hotspots of HTTP-requests in geographical maps. To do so, four different maps were modeled with pre-set hotspots. Nine algorithms were evaluated with a cost value to how well they could approximate the locations of the hotspots. Four of the algorithms were found to perform better than the other ones overall. Although the results does show potential in using hierarchical clustering algorithms, the low number of models makes it hard to determine the possibility for arbitrary maps of HTTP-requests.
Sammanfattning
Table of Contents

Introduction
  Purpose
  Problem Statement
Background
  Clustering
  Hierarchical Clustering
  Distance functions
  Linkage Criteria
  Other distance functions and linkage criteria
  Automation and radius interoperability
  User activity model
Method
  Model Request Map
  Algorithm Evaluation
  Clustering in Matlab
Results
Discussion
  Request Model
  Cluster consistency
  Cutoff value
  Cost results
  Algorithm improvement
Conclusion
References
Appendix A - Pseudo-code for Request Generation
Introduction

Place-finding services for smartphones are very popular today. Examples are Yelp, Foursquare and Google Maps. What they all have in common is that they provide information about nearby places given the user’s location. These places are often businesses and as such the information about them is often valid for a long time. However for dynamic information, such as news or job offers, the expiration date can be much shorter. Storing and updating such short-lived information is something that is inherently more difficult. In addition to the same functionality as for a place service, more consideration has to be taken into how long the information is valid and how often new information is produced.

An attempt at providing more dynamic information can be done with a location oriented crawler. The crawler gets information from a large set of information providers. The received data is stored in a database and later used by mobile clients. During a search iteration the crawler asks one or more providers for information related to an area. In specifying the area, a center point in geographic coordinates is given along with a radius. Different information providers allow for different search radiiuses and the crawler always uses the maximum radius allowed.

As there are restrictions in how often the crawler may ask for information there is a desire to keep the number of search iterations low while also providing relevant and up-to-date information to as many users as possible. Consequently, there are two things to consider before determining to ask a provider for information:

- The update frequency of the providers information in an area
- Which areas have higher levels of activity

The second consideration is what is focused on in this analysis, specifically how suitable hierarchical clustering algorithms are for finding these areas of high activity. The suitability is determined by how well an hierarchical clustering algorithm can find cluster centers given only a cluster radius preference from the crawler.
**Purpose**
The aim of this essay is to examine how suitable agglomerative hierarchical clustering algorithms are for identifying critical regions of activity, so called hotspots. An activity is defined to be a internet HTTP request being made from some point on earth. The purpose of finding these hotspots is to help a web crawler, a component of a search engine, prioritize regions to search for new or updated information. A suitable algorithm is one that can find hotspots, as center of a clusters, according to a circular size preference.

**Problem Statement**
Given a modelled map of randomly generated HTTP-requests weighted towards hotspots, how well can different agglomerative hierarchical clustering algorithms approximate the positions of these hotspots?
Background

Clustering
Clustering[2][3] is the act of grouping a set of objects such that two objects belonging to the same group are similar. Despite the concept of clustering being well defined, properties of an actual cluster is not. A cluster is much determined by the similarity measure used for grouping objects, which may be something that varies between applications. As new types of clusters are needed, new algorithms are developed, these often being extensions of existing algorithms.

There is no universal way or effective criteria to guide the selection of cluster features. In simplifying the work of finding a good algorithm it is important to look for features of the algorithms as well as properties of the desired clusters.

One difference between algorithms is the technique used for clustering. Algorithms using similar techniques are grouped into the same clustering model. Apart from the hierarchical model, the most common ones are the partitioning- and density based models. What differs these models from hierarchical models are the prerequisites necessary for the algorithms, besides the objects to be clustered. One example is \textit{k-means}, a partitioning algorithm, where a required input parameter is the value $k$ for how many clusters are to be found. In the density based algorithm DBSCAN inputs are the minimum number of elements per cluster and a density threshold.

In this thesis, the choice of hierarchical clustering algorithms is much because of the fact that they require no input data besides the data points to be clustered.

Hierarchical clustering

- An \textbf{agglomerative method} goes bottom-up, beginning with treating each sample object as a cluster and iteratively pairs two closest clusters to form a new one. The algorithm finishes once a single cluster remains.

- A \textbf{divisive method} conversely goes top-down, beginning with treating the whole set of objects as one cluster and iteratively dividing the clusters until all clusters contain one object.
An important difference between the methods is their time complexity: $O(n^3)$ for the agglomerative and $O(2^n)$ for the divisive method. For performance reasons the agglomerative is most commonly used[3], which is also the type used in this report.

The difference between hierarchical algorithms is which pair of linkage criteria and distance function that is used. The distance function is how to measure the distance between two individual points. The linkage criteria determines the distance as the distance function is applied to clusters containing more than one point each. The thesis will use the following distance functions and linkage criterias[1]:

**Distance functions**

**Euclidean distance**: $D(x, y) = \sqrt{\sum_{i=1}^{N} (x_i - y_i)^2}$

**Manhattan distance**: $D(x, y) = \sum_{i=1}^{N} |x_i - y_i|$  

**Chebychev distance**: $D(x, y) = \max \left( |x_i - y_i| \right), \forall i \in [1, N]$  

**Linkage Criteria**

**Complete linkage**: $d = \max \{D(x, y) : x \in C_1, y \in C_2 \}$

**Average linkage**: $d = \frac{1}{|C_1||C_2|} \sum_{x \in C_1} \sum_{y \in C_2} (D(x, y))$

**Weighted linkage (WGPMA)**: When clusters $A$, $B$ have been merged, the distance between $|AB|$ and any other cluster $C$ is: $d_{|AB|C} = \frac{1}{2} (d_{AC} + d_{BC})$. Distance at the bottom level is the distance function applied to the individual sample objects.
Other distance functions and linkage criteria

Some distance functions and linkage criteria are disregarded because they do not generate clusters that conform to the density or closeness strived for, or that they do not use distances measures that can be translated to from the crawler radius.

The distance functions are: Mahalanobis, Cosine similarity, Pearson correlation, Spearman correlation, Hamming and Jaccard.

The linkage criteria are: Centroid, Median, Single linkage and Wards method.

Dendrograms

A common way to illustrate the results of an agglomerative procedure is the dendrogram[1]. A dendrogram is a binary tree containing information about which pairings were made and at what distance a pairing occurred. As the dendrogram is computed only once, cluster extractions can be made several times on the same dendrogram. An extraction is done by setting a distance value at which to cutoff the dendrogram, ignoring any further clustering above that distance and return the cluster levels at the cutoff. The dendrogram is therefore re-usable for finding different clusters from one hierarchical clustering.

As the crawler varies its radius with the providers, the preference of the optimal cluster size varies. By clustering the HTTP request data, the returned dendrogram can be used for several provider radiuses. As this property of the dendrogram fits well with how the crawler works, the key to finding suitable hierarchical algorithms is by viewing how good the extracted clusters are when the cutoff is set to a search radius.

Figure 1.
A dendrogram resulted from a agglomerative clustering of three objects; A, B and C. The first pairing was A and B at distance $d_1$, the new cluster was then paired with C at distance $d_2$. A cutoff value between $d_1$ and $d_2$ would yield two clusters {A,B} and {C}. 
Geographic map model

User activity data may often be provided by businesses in percentage of usage according to user qualifications such as age or gender. However, geographical qualifications describing the location of a user's service request has not been found. For this reason we apply a way of generating these maps by using the positions of TELE2 cell cover towers for mobile internet traffic[4]. It is believed that the towers are placed optimally in providing mobile internet traffic where there are many users. The cell tower positions makes the hotspots for weighting the randomization of HTTP request points.
Method

Model Request Map

The request map was modeled as a two dimensional coordinate system with each potential request represented as a point in that coordinate system. Picking of points was done randomly, with an inclination towards predefined hotspots covering high frequency areas. Hotspots were selected based on positions of TELE2 3G cell towers in the city of Uppsala, Sweden.

The following algorithm was used to generate requests:

1. For every hotspot, set its coverage radius to $R$.
2. Create an array of points, $P$, containing all points in the coordinate system.
3. For every point $p \in P$ and number of towers, $T_p$, within distance $R$ from $p$:
   - re-add $p$ to the array $T_p \times F$ times, where $F$ is a predefined integral factor.
4. Randomly pick as many points as needed from $P$.

If $T(p)$ is the number of towers within distance $R$ of point $p$, the probability $\gamma(p)$ of picking any point $p$ in the coordinate system is hence defined by:

$$\gamma(p) = (F \times T(p) + 1) \times \left( |P| + \sum_{p \in P} F \times T(p') \right)^{-1}$$

Four named models were generated using the following parameters:

<table>
<thead>
<tr>
<th>Model name</th>
<th>Add factor $F$</th>
<th>Number of points</th>
<th>Hotspot radius $R$</th>
</tr>
</thead>
<tbody>
<tr>
<td>High Density/Few Outliers</td>
<td>100</td>
<td>5000</td>
<td>0.004</td>
</tr>
<tr>
<td>High Density/Many Outliers</td>
<td>2</td>
<td>10000</td>
<td>0.004</td>
</tr>
<tr>
<td>Low Density/Few Outliers</td>
<td>100</td>
<td>500</td>
<td>0.004</td>
</tr>
<tr>
<td>Low Density/Many Outliers</td>
<td>2</td>
<td>500</td>
<td>0.004</td>
</tr>
</tbody>
</table>

*Table 1.* The parameters used for generating the HTTP-request map models.
Algorithm Evaluation

The indicator as to how good the algorithms resembled the hotspots was evaluated with a cost value, based on the centroid of each cluster in the algorithms result.

The centroid of each cluster was calculated as the point in the cluster which minimizes the sum of the squared euclidean distance between itself and every other point in the cluster:

\[
C = \frac{p_1^2 + p_2^2 + \ldots + p_n^2}{n}, \quad p \in N,
\]

where \( C \) is the centroid and \( N \) is the cluster.

The cost of the algorithm was evaluated by:

\[
\sum_{c \in C} \min(\{ D(c, h), \forall h \in H \}),
\]

where \( H \) is the set of all hotspots and \( C \) is the set of centroids.

Clustering in Matlab

A Matlab script was used to generate and visualize the clusters. With input being all the generated HTTP points a conversion was made to a \( N \times 2 \) matrix, storing the \( N \) points row wise. With the matrix, the following Matlab functions were applied in order:

- **pdist**: Calculates the pairwise distance between each point in the matrix, using the specified distance function. Returns a proximity matrix.
- **linkage**: Forms a tree of the hierarchical clustering steps from a proximity matrix, using the specified linkage criteria. Returns a binary tree of clustering steps, meta-structure for a dendrogram.
- **cluster**: Constructs a list with cluster index for each point in the original set. This function accepts the cutoff value.

The cutoff value was set to \( 2 \times R = 0.008 \). All linkages were paired with every distance function, resulting in a set of nine configurations that each were ran once on all four models.
Results

The results are presented with four request maps. The red points in the maps are the pre-set hotspots, while the blue points are the randomly generated HTTP-requests. Each map is followed by a table showing the calculated algorithm costs for the map. The algorithms in the tables are ordered by cost in ascending order.
Figure 2.
The HTTP request model generated with properties High Density/Few Outliers.

<table>
<thead>
<tr>
<th>Linkage criteria</th>
<th>Distance function</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weighted</td>
<td>Chebychev</td>
<td>0.081565</td>
</tr>
<tr>
<td>Average</td>
<td>Chebychev</td>
<td>0.083422</td>
</tr>
<tr>
<td>Weighted</td>
<td>Euclidean</td>
<td>0.095537</td>
</tr>
<tr>
<td>Average</td>
<td>Euclidean</td>
<td>0.141303</td>
</tr>
<tr>
<td>Average</td>
<td>Cityblock</td>
<td>0.179241</td>
</tr>
<tr>
<td>Weighted</td>
<td>Cityblock</td>
<td>0.179562</td>
</tr>
<tr>
<td>Complete</td>
<td>Chebychev</td>
<td>0.226053</td>
</tr>
<tr>
<td>Complete</td>
<td>Euclidean</td>
<td>0.249952</td>
</tr>
<tr>
<td>Complete</td>
<td>Cityblock</td>
<td>0.404044</td>
</tr>
</tbody>
</table>

Table 2. Algorithm cost results for the High Density/Few Outliers model, ordered by the costs.
Table 3. Algorithm cost results for the High Density/Many Outliers model, ordered by the cost.
Figure 4.
The HTTP request model generated with properties Low Density/Few Outliers.

Table 4. Algorithm cost results for the Low Density/Few Outliers model, ordered by the cost.
Figure 5.
The HTTP request model generated with properties Low Density/Many Outliers.

<table>
<thead>
<tr>
<th>Linkage criteria</th>
<th>Distance function</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>Chebychev</td>
<td>0.132384</td>
</tr>
<tr>
<td>Weighted</td>
<td>Chebychev</td>
<td>0.147435</td>
</tr>
<tr>
<td>Average</td>
<td>Euclidean</td>
<td>0.155891</td>
</tr>
<tr>
<td>Weighted</td>
<td>Euclidean</td>
<td>0.167389</td>
</tr>
<tr>
<td>Weighted</td>
<td>Cityblock</td>
<td>0.253532</td>
</tr>
<tr>
<td>Average</td>
<td>Cityblock</td>
<td>0.266388</td>
</tr>
<tr>
<td>Complete</td>
<td>Chebychev</td>
<td>0.307277</td>
</tr>
<tr>
<td>Complete</td>
<td>Euclidean</td>
<td>0.325772</td>
</tr>
<tr>
<td>Complete</td>
<td>Cityblock</td>
<td>0.511374</td>
</tr>
</tbody>
</table>

Table 5. Algorithm cost results for the High Density/Few Outliers model, ordered by the cost.
Discussion

Each table in the result shows that there are four of algorithms that perform well consistently, compared to the other algorithms. The better ones are Average or Weighted linkage with Euclidean or Chebychev distance. It is tempting to state that these four algorithms are the most viable overall, but such claims require great caution and are in this case problematic. The primary reasons for this are that not enough tests has been made.

Request Model

Only one model has been generated per configuration. The name of the configurations describes a general point distribution, but the same configuration for generating the model could have yielded different points that were present. Because of this no guarantee can be made as to if the top algorithm is the best one overall, but only for the particular model. To improve the reliability of the results it is necessary to generate several models with the same configurations to catch as many variances in points selected as possible. In addition to generating more models with the same configuration, models have to be generated with different configurations. As real user activity data has not been found, there is a need for many models to be tested before a suitable algorithm can be determined for arbitrary clusters.

Evaluating the generated models is important for a correct cost measurement. Some models, such as the Low Density/Many Outliers will be sensitive to how the generated model actually looks. In the above case, there is a group of outliers that looks to be more coupled than the smallest cluster of a hotspot. Since the algorithms could regard this group of outliers as a cluster, there is a risk of getting a higher cost due to the modelling error. Although one could argue that all algorithms will have this additional cost, the problem of finding how erroneous it is for one particular algorithm is a different question altogether.

Cluster consistency

Validating cluster results is not a one-time process, it can be necessary to do a series of clustering repetitions[3]. This can be important for hierarchical algorithms since a pairing that has been made is not considered again. This can yield different results if several pairs of points share the same distance to each other. For providing reliable results it is therefore necessary to run the algorithm on the same model several times with the points in different order.
Cutoff value
The cutoff value has been set to represent a crawlers search radius in order to see how well the algorithms find clusters of that size given that the clusters do exist in varying degree. The suitability strived for has been to use the crawler search diameter as a cutoff value. Although this approach is more practical for its simplicity, it is still of value to consider alternative relations between crawler radius and cutoff value. Finding a right balance for each algorithm can reveal better results.

A comparison between the Complete-Cityblock and Weighted-Chebychev algorithms on the High Density/Few Outliers model can be made by viewing Figure 6 and Figure 7. Weighted-Chebychev has a lower cost due to it having closer distances to the hotspots while also not generating too many clusters. This is the opposite of the Complete-Cityblock algorithm. Complete-Cityblock instead has many smaller clusters around the hotspots, indicating a more small clustering in relation to the cutoff value. This hints that Complete-Cityblock could perform better if a larger cutoff value had been used.

Cost results
The cost algorithm results in better scores for algorithms that do not generate too many clusters and that can find centroids as close to the hotspots as possible. Since the points within a hotspot radius are distributed randomly it can be accepted to find centroids anywhere within the hotspot radius. As such a good result could be equal or less to 0.096, which is the hotspot radius times the number of towers. By this measure many algorithms has performed well as the best scores are often below the value.

Algorithm improvement
Due to the nature of hierarchical clustering, every point will be within a cluster. As such, outlier clusters might always be an additional cost for the algorithms. In finding the most critical hotspot areas, sorting the clusters by their number of points can be made. By finding relatively high drops in the number of points, less critical clusters could be ignored when the cost is calculated.
Figure 6. Hotspot resemblance for the Complete-Cityblock algorithm. The original hotspots are shown in red, and the cluster centroids as blue. The model used is High Density/Few Outliers.

Figure 7. Hotspot resemblance for the Weighted-Chebychev algorithm. The original hotspots are shown in red, and the cluster centroids as blue. The model used is High Density/Few Outliers.
Conclusion

There are four algorithms that consistently has been shown to have a low cost over all models:

- Average linkage with Chebychev distance
- Average linkage with Euclidean distance
- Weighted linkage with Chebychev distance
- Weighted linkage with Euclidean distance

However, for the problem statement no definite answer can be given. Further tests has to be made in order to provide more reliable results. These tests should involve a wider range of models, different models have to be generated with the same configurations as well as with new ones. In addition to new models, algorithms have to be tested repeatedly on single models to ensure consistency in their results. Improvements has been suggested to provide better and more accurate results for the algorithms. The improvements suggests removal of outliers clusters and finding better relations between the search radius and cutoff value.

Given that some algorithms do perform well on the models and that there is room for improvement, the conclusion is that there is potential in using hierarchical clustering algorithms for hotspot detection.
References


Appendix A - Pseudo-code for Request Generation

http_points ← List, initiated with every possible point
generated_points ← []
OCCURANCE_FACTOR ← Constant integer

// Re-add tower-vicinity points into http_points list
iterations = num(http_points)
for i = 0 → iterations :
    p = http_points[i]
    multiple = "number of towers near p"
    http_points.add(p, multiple*OCCURANCE_FACTOR)

// Randomly select points to represent HTTP-requests
for the number of HTTP points to be generated:
    p = http_points.get_random()
    generated_points.add(p)

return generated_points