Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

Improve Your Vacation

Semester thesis

Andreas Germann

agermann@student.ethz.ch

Distributed Computing Group
Computer Engineering and Networks Laboratory
ETH Zürich

Supervisors:
Pascal Bissig, Philipp Brandes
Prof. Dr. Roger Wattenhofer

February 26, 2016

Acknowledgements

I thank my supervisors Prof. Dr. Roger Wattenhofer, Pascal Bissig and Philipp Brandes for their support during my work on this project. Furthermore, I thank the ETH Zurich for providing the IT infrastructure I used in this work.

Abstract

We developed a program that automatically creates a crowd sourced tourist guide for a selected city. This tourist guide is to be based on the data of Flickr. To do this, the program analyses the meta data of geotagged photos from the image hosting platform Flickr.
These geotagged photos are clustered using DBSCAN to find the places with the highest density of photos. These high density places should coincide with places of interest in the city. Finally, these clusters are displayed on a map along with selected pictures from the respective cluster.
Four test cases are presented and evaluated. This evaluation showed that our algorithm was able to detect between 17 and 34 sights.

Contents

Chapter 1

Introduction

1.1 Motivation

Tourists that are going to visit a city need to know what there is to be seen in that city. To gather this information, they usually consult a tourist guide. In this work, we wanted to develop an algorithm that creates such a guide based on crowdsourced data from a social media platform.
Nowadays, social media platforms contain an abundance of information. This information is submitted by the users and is mostly unstructured. It may contain what a user likes, where he had been, what he had been doing and many more things. Data analysis algorithms can be used to derive further and more general information from this plethora of individual data.

For our work, we use Flickr as the source of this information. Flickr is a social media platform that allows people to share their photos with others. From there, we collect the metadata of geotagged photos. Using these geotagged photos, we can see where Flickr's users have taken photos. Following the assumption that a high number of photos indicate that there is something interesting to be photographed, we derive the points of interest from the distribution of these photos. Furthermore, we want to improve these tourist guides by including photos to advertise the sights we found. These photos are taken from the set of photos that made up the high density area. The selection is made based on a photo rating of Flickr's website.

1.2 Approach

For this analysis, we first crawl the geotagged photos in a city from Flickr. These photos are then analysed to find high density areas which should contain a point of interest. For these areas, a number of photos is selected which should represent the sight that is there. Finally, a map is created to show the location of the

discovered sights along with the pictures.

1.3 Related Work

Our work is related to the work on recommender systems. Recommender systems try to predict a users preferences in order to make recommendations to him. In particular, our work is related to recommender systems based on collaborative filtering. This means that the recommendations are made based on data gathered in observations of large sets of similar users. [1]

One example for for this can be found on the webshop Amazon.com™. For each item in their shop, they make suggestions for other products based on the analysis of what other customers bought together with the item.
Another example for this can be found on the online auction website eBay.com™. There, sellers and buyers can rate each other after a transaction. These ratings can be used by other users to determine the trustworthiness of another user. [2]

In our work, we try to predict the interest of a tourist in a place. This prediction is based on the interest of Flickr's userbase in a certain location.

Chapter 2

Data Acquisition

This chapter explains what steps were involved in the acquisition of the data necessary for the analysis. The data was acquired using Flickrs API, a powerful tool provided by Flickr. The first step was to create a bounding box for the city that was to be processed. A bounding box is a rectangular area, defined by a tuple of coordinates. The second step was to crawl Flickr for the *PhotoIds* of the pictures in this bounding box. The last step was to acquire the full metadata of the photos from Flickr. This was very time consuming and scaled linearly with the number of photos. Therefore, a reduction of the datasets based on the user analysis was made beforehand. Furthermore, this user analysis required the acquisition of data about the users.

2.1 Flickr API

Flickr, as the source of our data, offers a sophisticated API for their database. It allows almost all operations a user can perform on the website. For our case, the methods for querying the database are especially interesting. The API is easily accessed in a lot of programming languages like Java, Python, Ruby, etc.

For this work, certain parts of Flickr's meta data were needed. The *PhotoId* uniquely identifies every photo in Flickr's database. The *UserId* identifies the user, that uploaded a photo. *Latitude* and *Longitude* represent the geotag. *Taken* gives the date and time, a picture was taken. And *Favorited* gives the number of people that have marked this photo as a favorite, similar to the *Like* functionality on Facebook. Furthermore, there are some administrative variables. These consist of a *ServerId*, a *FarmId* and a *Secret*. The *ServerId* and the *FarmId* describe the location of the photos file in Flickr's server system and the *Secret* is part of Flickr's access control mechanism. These were needed for downloading the picture, along with the *PhotoId*.

Queries on the Flickr database have to be sent to the Flickr servers, processed and then sent back. Therefore, each query takes about half a second. For the analysis mentioned in Chapter 3, such a time delay was not convenient. Therefore, the needed data was queried and stored in a local MySQL database beforehand.
This time delay was also a problem in the acquisition of the full meta data. Since two API calls had to be made for every photo separately, the acquisition of the full meta data took a very long time (see Section 2.4).

Flickr wants to prevent users from querying large parts of the database at once, because such queries lead to a high peak load on their database. Therefore, Flickr has some restrictions in their API. One of them considerably increased the effort of acquiring the data we needed. This is explained in detail in Section 2.3.

2.2 Bounding Box

The first step was to create a bounding box for the city that was to be processed. A bounding box is an object that represents a rectangular area. It consists of the minimum and maximum longitude and latitude. This minima and maxima represent the boundaries of the rectangle. In this case we wanted the rectangle that encompasses the city's area. These bounding boxes were later used as boundaries for the geotags while crawling photos of a city.

Flickrs uses a tree like system for spatial organisation called *PlacesId*. This system consists of the levels *Continent*, *Country*, *Region*, *County*, *Locality* and *Neighborhood*. It is possible that a level is skipped in this tree. Figure 2.1 shows an example for such a hierarchy.

In this part, the methods *flickr.places.getByLatLon* and *flickr.places.getInfo* were used.
flickr.places.getByLatLon returns the *PlacesId* of the place in the lowest available level for a given set of coordinates. This was usually on the *Locality* or *Neighborhood* level.
flickr.places.getInfo returns information on the place specified by a *PlacesId*. This information includes the parent nodes on all higher levels as well as geographical information on the place.

First, a geocoder was used to get the coordinates of the city that was to be processed. With *flickr.places.getByLatLon* the *PlacesId* for this place was called. Then, the parent node on *County* level for this *PlacesId* was looked up using

flickr.places.getInfo. If no parent node on the *County* level existed, the next higher level was checked until a parent was found. Usually, either a *County* or a *Region* parent node existed. This process is shown in Figure 2.1 with the red arrows.

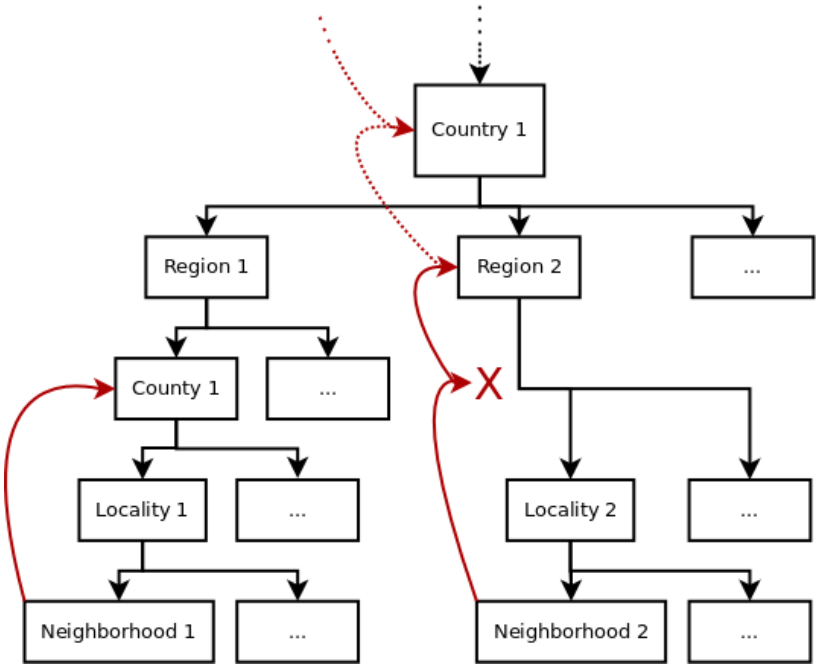

Figure 2.1: Graphical representation of the Flickrs place hierarchy; the red arrows indicate the search for a parent node

For this parent node, *flickr.places.getInfo* was used to get its geographical information. This geographical information consists of a list of coordinates. This list represents the corners of a polygon that describes the spatial extent of this place. From this polygon, the bounding box was created by taking the maxima and minima of latitude and longitude out of this list. This bounding box is used in Section 2.3 to query the *PhotoIds* in this city. The polygon itself was not used, because the rectangular bounding box can be divided easier. This will become necessary in Section 2.3.
Figure 2.2 shows the bounding box for Zurich and how it was created from the polygon.

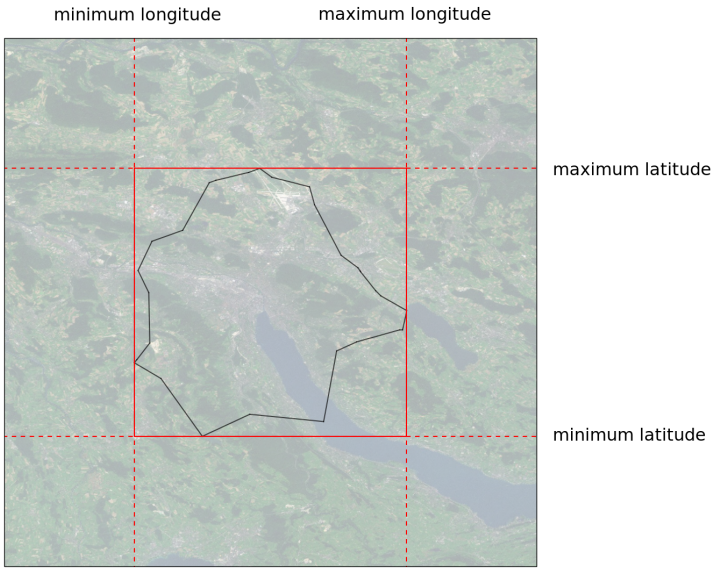

Figure 2.2: Polygon for Zurich (black) and the resulting bounding box (red); map data from arcgisonline.com [3]

2.3 Photo Id

The next step was the acquisition of the *PhotoId* and the *UsedId* of all photos in this bounding box.

To do that, *flickr.photos.search* was used. *flickr.photos.search* takes a wide range of criteria as input. The criteria that we used were the date and location a picture was taken and the user that took the picture. The function returns a list of all photos that meet these criteria.

At this point, the restriction became noticeable. For every query, Flickr only answers with 4'000 photos. While Flickr still answered with the correct total amount of pictures, the list of photos repeated after the 4000^{th} entry. To get more photos, the query needed to be further specified.
Therefore, the crawling algorithm was designed recursively. The bounding box was quartered if there were more than 4'000 photos in it and the method recalled on every quarter. This was done until less then 4'000 pictures were in the bounding box. Then, the list was read out and stored in the local database.
Another problem was the precision of longitude and latitude. When the division of the bounding box reached this precision limit, the recursion had to be continued on another value. The date the photo was taken was chosen for that matter.

This recursion started with the interval from 1.1.1970 up to the current date. Every time an interval contained more than 4'000 pictures, it was halved and the function called on both halves. When an interval had less than 4'000 pictures, these were queried from Flickr and stored in the local database.
Figure 2.3 shows the principle of how a bounding box was divided. This recursion is shortened for simplicity reasons. The real algorithm reached much higher recursion depths. The recursion by time worked similarly. The difference was that it only had one dimension.
This yielded a local database of all photos in the city. It contained the *PhotoIds*, the *UserIds* and the administrative variables of these pictures.

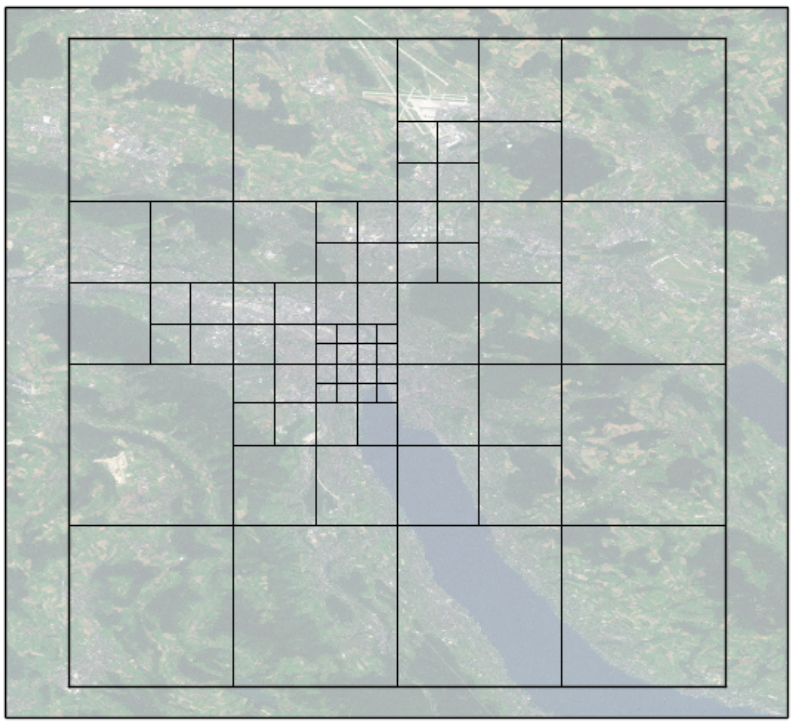

Figure 2.3: Bounding box of Zurich, showing the principle of the spatial recursion; map data from arcgisonline.com [3]

flickr.photos.search responds in pages of up to 250 elements. This means that with one query 250 photos could be crawled. Because of this, these values could be queried within a few minutes or hours depending on the number of photos in the city.

2.4 Full Metadata

After the collection of the *PhotoIds*, the remaining meta data had to be collected.

This was done using the methods *flickr.photos.getInfo* and *flickr.photos.getFavorites*. *flickr.photos.getInfo* returns the metadata for a photo specified by a *PhotoId*. *flickr.photos.getFavorites* takes a *PhotoId* as input and returns a list of the users that have made the picture one of their favorites.

flickr.photos.getInfo and *flickr.photos.getFavorites* had to be called for all the *PhotoIds* in the local database. Since these two functions had to be called on every photo separately, this took about one second per picture. Because of this rate, the acquisition of the full metadata of all the photos in the datasets would have taken several weeks for the big cities such as Paris or Seattle (see Table 2.1). Therefore, the datasets had to be reduced first (see Section 2.6).

2.5 User Data

This dataset reduction was made based on a user analysis. For this user analysis, data about the users was needed. Therefore, the total amount of pictures a user had uploaded was queried for every user in the local database. Furthermore, photos of every user were counted in the local database. These numbers were later used to calculate the ratio between a users total photos versus his pictures in the city. In Section 3.1 this ratio is used to reduce the datasets based on a user analysis.

This total number of picture of a user was queried using *flickr.photos.search*. This returns a list of photos that match the input criteria. In this case the criterion was a *UserId*. The elements of the returned list were counted and gave the resulting total number of pictures of the specified user.

2.6 Reducing datasets

As mentioned in Section 2.4, the acquisition of the full metadata was a very time consuming process. For every photo, two API calls had to be made. This lead to an execution time of one second per photo or several days for an entire city. To speed this up, the datasets had to be reduced. Table 2.1 shows the characteristics of the four available datasets. It shows the total number of photos in

a city, the number of photos after the first reduction and the number of photos after the second reduction. Furthermore, it shows the estimated time effort for the acquisition of these numbers of pictures.

City	Paris	Seattle	Singapore	Zurich
Total photos	2356961	1578384	824780	192062
Photos after 1st reduction	1178497	793088	412998	96389
Photos after 2nd reduction	501629	504509	412998	96389
Acquisition time for all photos	27.3d	18.3d	9.5d	2.2d
Acquisition time after 1st reduction	13.6d	9.2d	4.8d	1.1d
Acquisition time after 2nd reduction	5.8d	5.8d	4.8d	1.1d

Table 2.1: Reduction of photos and acquisition time

The first reduction was made based on the user analysis mentioned in Section 3.1. In that step, the datasets were reduced by 50%. The process for this reduction is explained in detail in that section. This was done with all sets even though some sets were already smaller than the limit that will be set later. That was done, because it also served the purpose of filtering out non-tourist users.
But this reduction proved to be insufficient. For example, the reduced dataset of Paris with more than a million photos would still take about 13.6 days. And there are cities with even more photos. For example, London has more than five million.

Therefore, a limit of 500'000 photos was introduced. This limit can be exceeded because of the way the algorithm works (see Section 3.1). Photos are added based on a user selection and the photos of a user are added as a whole. Therefore the photos of the user that is added last can exceed the limit.
This limitation kept the acquisition time under six days. The limit was set without further reasoning aside from the time effort.

Chapter 3

Data Analysis

In this chapter, the algorithms for analysing the data are discussed. The first part is about the reduction of the datasets mentioned in Section 2.4. The second part is about *DBSCAN*, the chosen clustering algorithm. The third part is about the parameters of *DBSCAN* which define how strict the clustering is. These parameters were important for a successful recognition of the points of interest. The last part describes the second clustering which clustered the points of interest from the first clustering again. This was done to group clusters together that came from one big sight but that was split into multiple clusters.

3.1 User Analysis

As mentioned in Section 2.6, the datasets had to be filtered, since a lot of non-touristy photos were in it. Results of early tests showed that some clusters were not near any points of interest. Closer inspection revealed that these clusters represented local users and even web presences of companies. Since the goal was to find tourist hotspots, these users had to be filtered out.

This filtering was made by calculating the ratio between the pictures a user had taken in the city and the total number of pictures on his account. Local users and businesses were considered more likely to have a big part of their photos in the respective city, since they spend most of their time there. Tourists on the other hand would have small percentage of photos there, since they only visit the city for a short period of time. The photos to be used were selected by adding the pictures of users one by one in ascending order of this ratio until 50% of all the pictures in the city were added. This solution yielded the pictures of users that were most likely tourists.
In addition to the user analysis, this method reduced the amount of pictures and therefore the acquisition time by 50%.
The additional limit of about 500'000 photos per city mentioned in Section 2.6

was included in this method. Photos were added, until 50% of all pictures or 500'000 were added, whichever happened first. The photos of the last user were added as a whole even if they exceeded the limit. That is why there can be slightly more than 500'000 photos after the second reduction.

Table 3.1 shows the effect these filterings had on the datasets of Paris, Seattle, Singapore and Zurich. It shows the reduction of photos and users in total numbers and proportionally.

City	Paris	Seattle	Singapore	Zurich
Total photos	2356961	1578384	824780	192062
Selected photos	501629	504509	412998	96389
Share of photos selected	21.3%	32.0%	50.1%	50.2%
Total users	64917	26548	16977	6973
Selected users	40340	21552	14814	6032
Share of users selected	62.1%	81.2%	87.3%	86.5%

Table 3.1: Reduction of photos and users in total numbers and in proportions

The table shows that while the total amount of photos was reduced by 50 to 80%, the number of users was only reduced by 10 to 40%. This shows that even after these reductions, the photos still represent a large part of the userbase.

3.2 Clustering Algorithm

Having created the datasets, the next step was to select a clustering algorithm. This choice was made based on the dataset of Zurich, since it was the first one available. For the recognition of patterns in the dataset, a ground truth of the sights in Zurich was needed. This was taken from the Wikitravel site of Zurich. [4] Wikitravel is a website that contains crowdsourced tourist guides.
The raw data for the center of Zurich can be seen in Figure 3.1 (left). Every photo is represented by a black dot with an opacity of 0.02. Therefore, darker spots indicate places of a high photo density. The yellow stars indicate sights that are listed on Wikitravel.
These figures show that the points of interest correlate strongly with high density areas. This fits the reasoning that areas around sights will have a high photo density, mentioned in Section 1.1. Therefore, the clustering algorithm should be density based. Furthermore, the algorithm should allow for samples that are not in any cluster, since there can be photos not linked to any sight.

Figure 3.1: Map of raw data (left) and clustered data (right) in the center of Zurich; stars represent sights; map data from arcgisonline.com [3]

For these reasons, ***D**ensity-**B**ased **S**patial **C**lustering of **A**pplications with **N**oise* was chosen as our clustering algorithm. *DBSCAN* is density-based and allows samples outside of clusters, in contrast to other algorithms like for example *k-means* [5].
DBSCAN takes two parameters, ε and *minSamples*. A dense region is defined as a circular area of radius ε with at least *minSamples* samples in it. The algorithm starts with a random sample and checks, whether it is the center of a dense region. If it is, a new cluster is started. If it is not, it is considered noise. But might belong into the dense region of a different sample.
In case a new cluster is started, all samples in its ε-neighborhood are added to the cluster, as well as their ε-neighborhoods if they are dense, too. This is continued recursively until samples with non-dense neighborhoods are reached. Then another previously unvisited sample is chosen and the process starts again until all samples have been visited. [6]

An example for this process is shown step by step in Figures 3.2 to 3.5 with $minSamples = 4$.

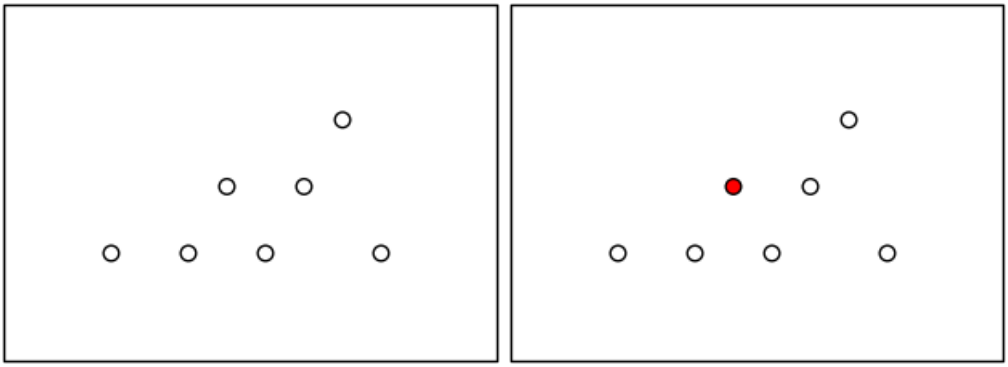

Figure 3.2: Unordered set at the beginning (left); choose a sample to start with (right)

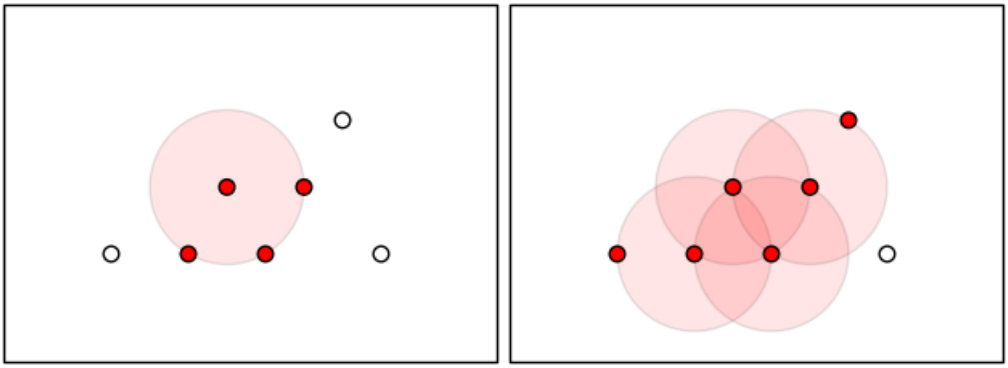

Figure 3.3: Check its ε-neighborhood for samples, if enough are found, start new cluster (left); insert samples from neighborhood into cluster and check their neighborhoods (right);

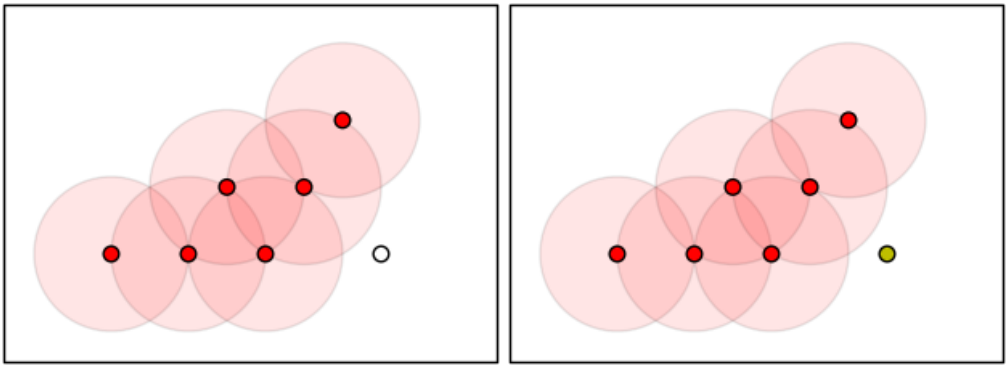

Figure 3.4: Repeat until no more samples are added (left); choose another unvisited sample (right)

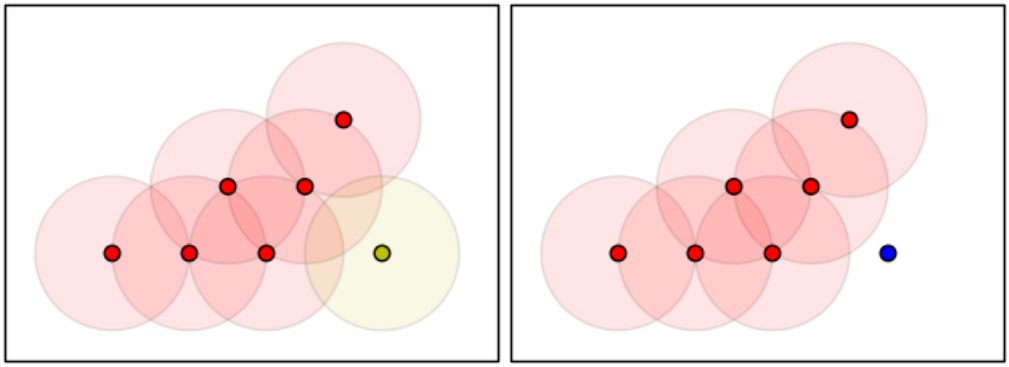

Figure 3.5: Check its neighborhood for samples (left); if not enough are found, mark as noise (right)

3.3 Clustering Parameters

After choosing *DBSCAN* as algorithm, its parameters ε and *minSamples* had to be determined. Because of the varying total number of photos, *minSamples* was defined as a percentage of all photos. Clusterings with different parameters were made on the dataset of Zurich. For these clusterings, we identified values like the total number of clusters, the number of sights found from a predefined list and the maximum number of sights, that merged into one cluster. Their results are shown in Figures 3.6, 3.7 and 3.8.

The total number of clusters was considered important, since these clusters would finally be the suggestions made to a user. The goal for this number was set to 40 to 60. 60 as an upper limit, since a user will probably not want to go through hundreds of suggestions. 40 as a lower limit, since the user should be presented with a selection to choose from. Also, 40 to 60 would be our personal preference.
The number of discovered sights was considered, since it was the goal of the program to find such sights. The goal was a number of discovered sights as high as possible. The sights listed on Wikitravel served as the ground truth. [4]
The last value was the number of sights from Wikitravel that were merged together into one cluster. In such a merging, a small sight might be lost in a bigger one. This happens if after a merging, only pictures of the bigger sight are shown. The method of picture selection we used could not prevent this. In that case a user will not get any indication that there is another sight in that cluster and might ignore it. Therefore, the number of merged sights should be as low as possible. In Figure 3.1 such mergings can be seen.

ε showed to be closely linked with the size of the clusters and their mergings. Large ε lead to large clusters and therefore mergings. Furthermore, a large ε can reduce the number of clusters by merging. Furthermore, we found out that *minSamples* has a big influence on the number of clusters. A small *minSamples* leads to a lot of clusters and in turn also to a lot of found sights. These correlations can be seen on Figures 3.6, 3.7 and 3.8.

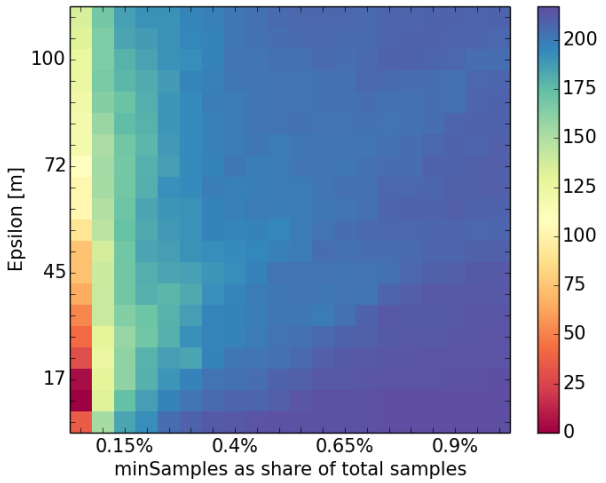

Figure 3.6: Number of clusters in Zurich depending on ε and *minSamples*

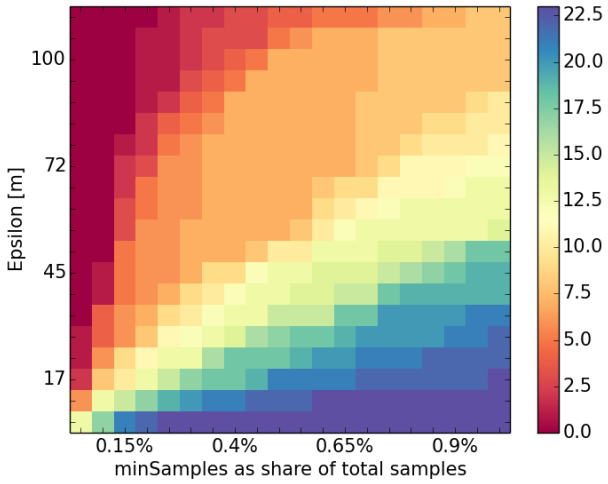

Figure 3.7: Number of sights found in Zurich depending on ε and *minSamples*

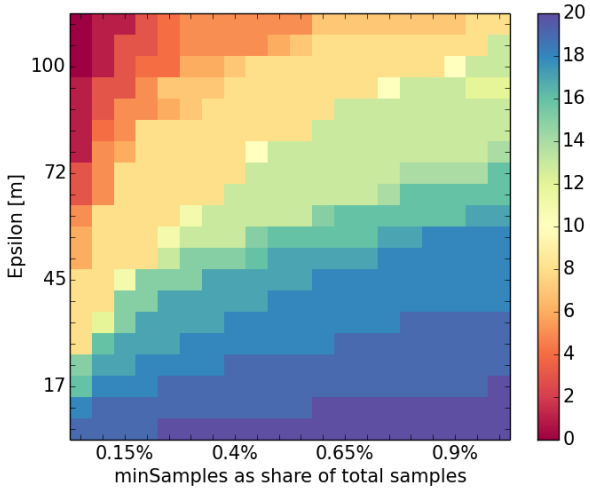

Figure 3.8: Maximum number of sights merged together in Zurich depending on ε and *minSamples*

Based on the results for Zurich, ε was set to 27 meters and the percentage of *minSamples* was set to 0.15% of the total samples. This produced 59 clusters which matched 17 out of 25 sights from Wikitravel with a maximum of three sights merged into one.
But applying these parameters to Singapore only yielded 16 clusters. With this number being far below the targeted 40 to 60 clusters, the algorithm was changed to be dynamic.

In this dynamic solution, it was decided to keep one parameter fixed, to reduce the two-dimentional problem into a one-dimentional one. ε was kept at 27 meters. ε was chosen to be static, because it produced clusters of about 200 to 350 meters in diameter in both Zurich and Singapore. This diameter was considered a good cluster size, since it is about the size of a sight in reality. The only problem were exceptionally large sights like parks or zoos. This problem is addressed in Section 3.4. Furthermore, *minSamples* had shown to be more serviceable for controlling the number of clusters (see Figure 3.6).

Therefore, the percentage for *minSamples* was determined automatically for every city. The algorithm would create multiple clusterings with different values until a valid number of clusters were created. The algorithm started with *minSamples* at 1% of the total samples and made a clustering. Everytime, less than 40 clusters were created, *minSamples* was reduced by 20% and a new clustering was made. If more than 60 clusters were created, *minSamples* was increased by 20% and a new clustering is made. If between 40 and 60 clusters were produced,

this clustering was used. For better understanding, pseudo code for this algorithm is shown in Algorithm 1.

```
epsilon = 27 m;
minSamples = 0.01 * totalSamples;
loop:
clustering;
if number of clusters is between 40 and 60 then
    go to finish;
else
    if number of clusters is smaller than 40 then
        minSamples = minSamples * 0.8;
        go to loop;
    else
        minSamples = minSamples * 1.2;
        go to loop;
    end
end
finish:
```

Algorithm 1: Dynamic determination of *minSamples*

This algorithm did not give clusters above a certain density, but the 40 to 60 most photographed spots. That way, the targeted number of clusters were reached by finding the most interesting places in the city.

3.4 Second Clustering

As mentioned in Section 3.3, the selected clustering parameters would still split up larger sights, for example historic centres, parks or zoos. To address that a second clustering was introduced. This second clustering was performed on the points of interest found in the first clustering. For this clustering ε was set to 300 meters and *minSamples* was set to 1. That way sub-clusters within walking distance of each other are grouped together with these groups representing the larger sights.

These groups were later to be presented in the same color to show their linking, but with the individual sub-clusters still present. That way, a smaller sight could not be lost within a bigger one, while large sights were still presented as a group. An example is shown in Figures 3.9 and 3.10.

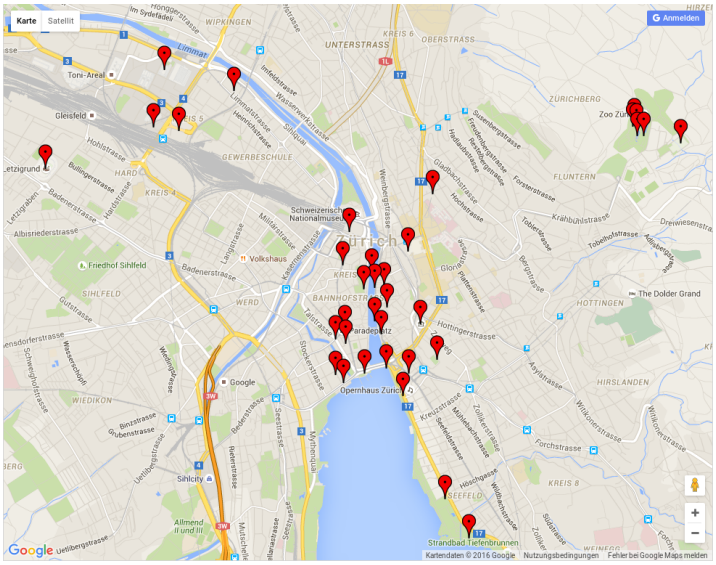

Figure 3.9: Points of interest in the center of Zurich without 2nd clustering; map data from Google Maps [7]

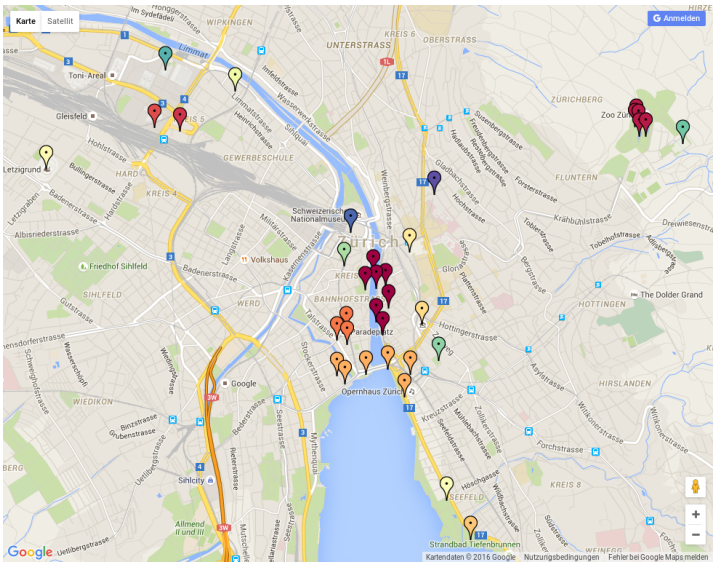

Figure 3.10: Points of interest in the center of Zurich with 2nd clustering; map data from Google Maps [7]

Chapter 4

User Interface

In this chapter, we discuss how the discovered points of interest are presented to a user. The user interface was implemented as an interactive map. This map shows the location of the clusters along with pictures from the cluster to give a reference of the sight in it. The first section explains the selection of these pictures. The second illustrates the implementation of the map itself.

4.1 Selection of Pictures

The potential points of interest that were found in the data analysis were now to be presented to a user. To do that, a selection of pictures from every cluster was chosen. Ideally, these pictures would show the sight that is in this cluster, like a landscape or a building.

In a first attempt using a random selection, a lot of pictures were not useful for representing the sight in their cluster. Some were focused on people and only contained the sight in the background. Others were pictures of things unrelated to the actual sight. These pictures are not useful for representing the point of interest in the cluster. Furthermore, certain pictures containing people are in a legal gray area because of privacy laws. For these reasons, we tried to keep such pictures out of the selection.

Figure 4.1 shows two examples for such unsuitable pictures from the cluster at Polyterrasse in Zurich. The person in the example was pixelated for privacy reasons. Figure 4.2 shows an example of a representative picture for the same cluster.

The only available indication for the quality of a photo was the *Favorited* value. This is the number of people that had favorited a photo on the Flickr website (see Section 2.1). Therefore, *Favorited* was used as a measurement of

Figure 4.1: Non-representing pictures from the cluster at Polyterrasse in Zurich with unrelated motives (left) [8] and with a person (right) [9]

Figure 4.2: Representative picture from the cluster at Polyterrasse in Zurich [10]

how good a photo is. The reason behind it was that if a lot of people favorited a photo, it should be good. And good pictures in the vicinity of a sight were considered more likely to contain the sight. Therefore, the pictures were selected in descending order of their *Favorited* value.
This solution improved the quality of the photo selection compared to a random selection. But it did not guarantee that the sight was on the pictures, since the users do not exclusively rate a picture based on this standard.

To increase the chances of selecting at least one useful picture, the selection should be as big as possible. In contrast to that, the number of pictures needed to be kept within certain limits, because a user should not be confronted with too many pictures. Therefore, the number of pictures per cluster was set to eight.

The results from the test cases showed that the performance of this selection method still had room for improvement (see Section 5.3). A method to further improve this selection could be a graphical analysis of the photo itself. But this was out of the scope of this work.

4.2 Website

In order to present the discovered points of interest to a user, a website was created for every processed city. These sites contain an interactive map displaying the discovered sights along with the selected pictures from Section 4.1.

This was implemented using the Google Maps API. With this API, a user can create an application based on Googles map service. For example, markers in different shapes and colors can be set at specified locations. And info windows can be attached to such markers to display further information about this location like a text or pictures.

In our case, markers were set at the locations of the found clusters. These markers show where potential points of interest are. Attached to those markers are info windows which open when a user clicks on the marker. These info windows contain the selected pictures of the respective cluster.
Different colors were used to represent the groupings of the second clustering mentioned in Section 3.4. A different marker color was used for every super cluster.

As an example, Figure 4.3 shows the website of Paris with an info window opened.

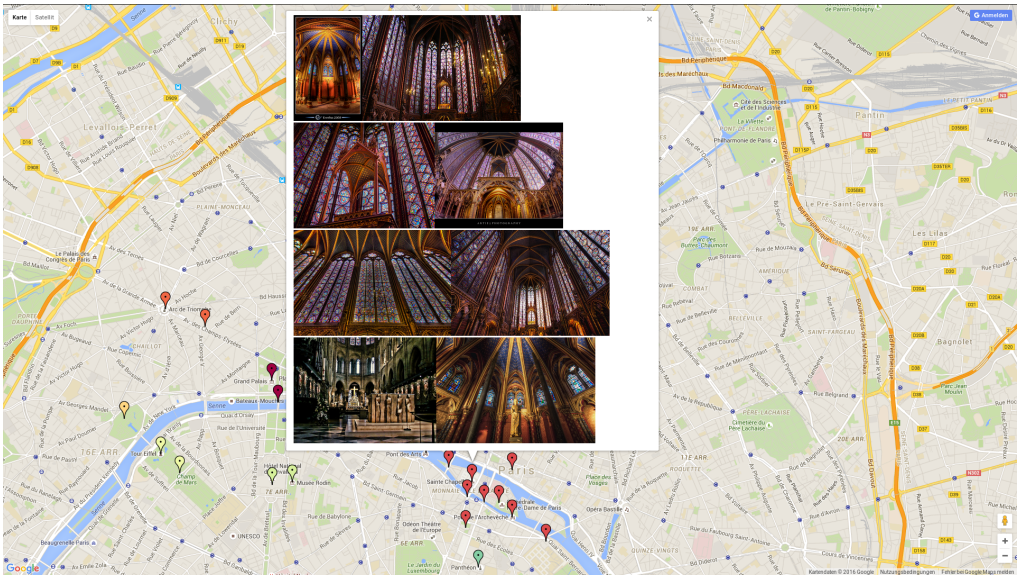

Figure 4.3: Interactive map of Paris with an open info window; map data from Google Maps [7]; photos [11, 12, 13, 14, 15, 16, 17, 18]

Chapter 5

Evaluation

In this chapter, we evaluate the algorithm that resulted from this work. This evaluation is made based on the four test cases Zurich, Singapore, Seattle and Paris. The results of the algorithm for these cities are compared to the ground truth taken from Wikitravel. [4, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68] Wikitravel is a website that contains crowdsourced tourist guides.

This evaluation is made based on the clustering algorithm, the second clustering and the picture selection.
The goal of the clustering algorithm was to find sights in a city. This algorithm is evaluated by comparing its results to the sights from the ground truth.
The second clustering was supposed to group the results from the first clustering. These groups would represent larger sights that were split up by the first clustering.
The goal of the picture selection was to choose eight photos for every cluster in order to represent it. This selection is evaluated by analysing the quality of these pictures.

5.1 Clustering Algorithm

The performance of the clustering algorithm is based on two values. The first one is the number of sights from the ground truth that the algorithm successfully discovered. The second one is the number of clusters that could not be explained by any nearby sight. These clusters were considered erroneous.

Note that the number of successfully discovered sights and the number of erroneous clusters do not necessarily add up to the total number of clusters. This is because one sight might be represented by multiple clusters or multiple sights

might be represented by one cluster.

City	Paris	Seattle	Singapore	Zurich
Number of clusters	40	49	40	46
Number of sights on Wikitravel	116	99	99	26
Number of sights found	34	33	29	17
Number of erroneous clusters	2	5	6	3

Table 5.1: Evaluation of the clustering in the four test cases

These results showed that the number of detected sights was between 28 and 44. Furthermore, the number of erroneous clusters was between two and six. The ratio between erroneous clusters and successfully discovered sights lay between 1 to 20 and 1 to 6.

Looking at the four test cases, it can be seen that the algorithm performed comparably in all the cities. Note that the datasets of Paris and Seattle were reduced by 68% and 78% while Zurich and Singapore were each only reduced by 50%. This might suggest that a more extensive dataset reduction can be done without degrading the final result.

5.2 Second Clustering

Figures 5.1 and 5.2 show maps of the four test cases. The maps show the centers of the respective cities, because they are the place where the second clustering can be seen best.

In Singapore, examples for such groupings are the area around the Marina Bay Sands, the Orchard Centre or chinatown. The grouping of the area between the National Museum, and the Suntec Convention center is an example of an unnecessary grouping.
Examples for Seattle are the area around the Space Needle, the Olympic Sculpture Park or the piers around the Waterfront Park.
Good examples in Paris are Montmartre, the Eiffel Tower or Versailles. The grouping of the clusters around at the Ile de la Cité and along the Seine has a length of 2.5 kilometers. Therefore, we considered it too big.
In Zurich, good examples are the lake promenade, the zoo or the historic centre along the Limmat.

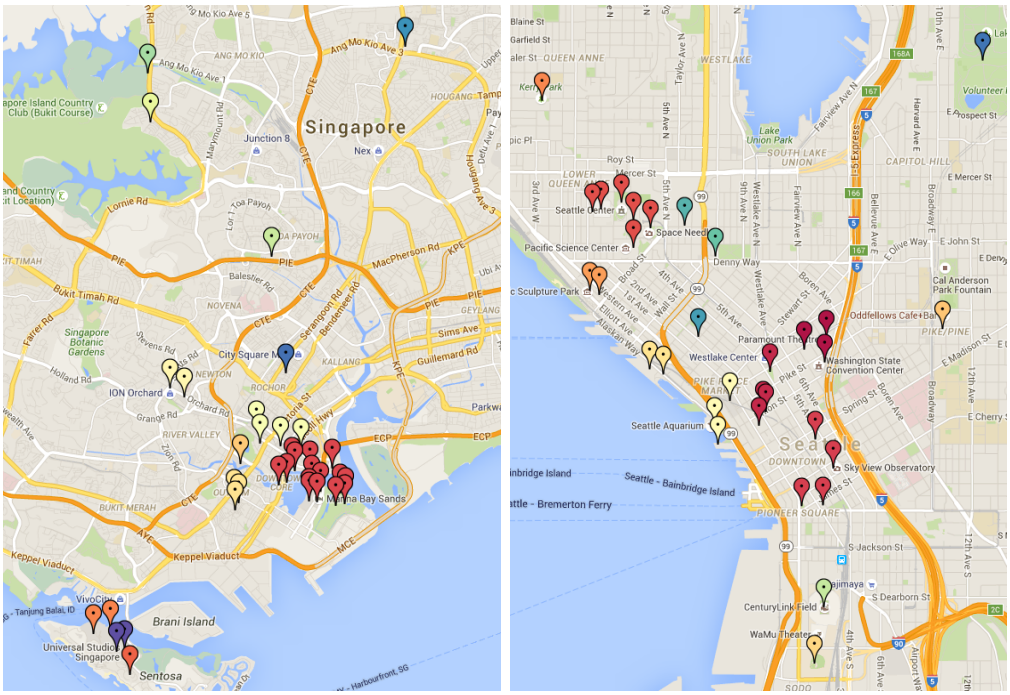

Figure 5.1: Singapore (left) and Seattle (right) after 2nd clustering with groupings represented in the same color; map data from Google Maps [7]

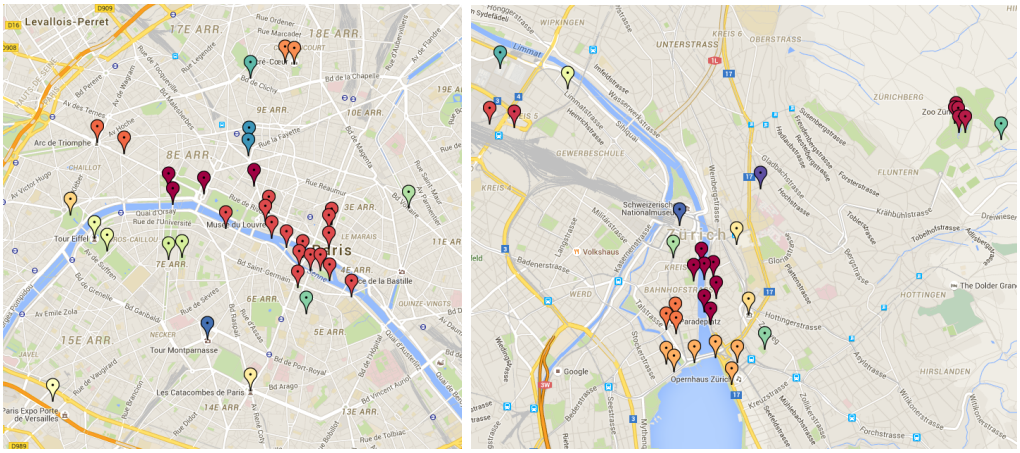

Figure 5.2: Paris (left) and Zurich (right) after 2nd clustering with groupings represented in the same color; map data from Google Maps [7]

5.3 Photo Selection

For the evaluation of the photo selection, we analysed the photos and divided them into four categories. Into the first category came photos that were representative for their cluster. These were considered good. The second category consisted of photos that mainly showed people. These were considered bad, since they have the focus on the people and contain the sight only in the background. The third category contained photos that showed things unrelated to the sight. These were considered bad because they are not representative for the cluster. The last category were photos with misplaced geotags. These were considered bad because they can contain a good photo of a sight but present it in the wrong place.

Table 5.2 shows the numbers of these categories for Paris, Seattle, Singapore and Zurich.
Note that the number of total photos are not always eight times the number of clusters. This is because some photos were not downloaded correctly or were not available any more.

City	Paris	Seattle	Singapore	Zurich
Total number of photos	316	386	312	364
Representative photos	241	216	143	230
Photos with people	29	89	63	59
Unrelated photos	37	80	75	63
Misplaced photos	9	1	31	12

Table 5.2: Evaluation of the photo selection in the four test cases

This analysis showed that the quality of the picture selections depended on the city they were for. In the test cases, the share of useful pictures was between 45% and 77%. Furthermore, the percentage of the pictures with people was between 9% and 24%. And the share of misplaced photos was between 0.3% and 9.9%.

Looking at this numbers, we thought that a graphical photo analysis could improve the quality of the selection. In such an analysis, face recognition could be used to filter out photos with people in it. Furthermore, such an analysis could help with detecting misplaced geotags. But the implementation of a graphical picture analysis was out of the scope of this work.

Chapter 6

Future Work

The evaluation showed that the algorithm developed in this work still has room for improvement.
A first issue was the time effort for the data acquisition of several days or weeks. This was lessened by a dataset reduction but not resolved completely. The evaluation in Chapter 5.1 showed that the different cities yielded comparable results even though some of them were reduced more heavily than others. Therefore, further reduction could be a part in future work.
The second problem was the selection of pictures for representing a discovered sight. The selection based on the number of users that favorited a photo often yielded mixed results (see Table 5.2). An analysis of the picture itself could improve this selection. This could be implemented and evaluated in future work.

6.1 Dataset Reduction

In this work, the acquisition of the full metadata of all the pictures took several days for the bigger cities. This was mitigated by a reduction of the datasets to 50%. For very big cities, an additional limit of about half a million pictures was introduced. These reductions limited the execution time of the data acquisition to about six days (see Section 2.6).

The evaluation in Section 5.1 revealed that the clustering algorithm performed comparably in all test cases. Despite the fact that they were reduced to different extents. For example, the dataset of Zurich was reduced to 50% of its initial size while the one of Paris was reduced to 20%. Still, the results of the two cities were of similar quality (see Table 5.1).

This leads to the question, how far a dataset can be reduced before the quality of the result suffers too much. An optimal extent of reduction could minimize the acquisition time while still yielding a good result. Finding such an optimum

could be the subject of future work.

6.2 Graphical Picture Analysis

As mentioned in Section 4.1, photos are needed for the presentation of the discovered points of interest. These pictures would ideally show the sight. But a lot of pictures included people and the sight only in the background. Other pictures were not representative because they showed things unrelated to the sight. The selection based on the number of users that had favorited a photo improved the selection compared to a random one. But this selection still contained a considerable amount of unsuitable pictures (see Table 5.2).
Furthermore, it occurred that photos showing a certain sight were shown in a different cluster. These errors were caused by misplaced geotags and therefore showed up at the wrong place. This problem was especially severe in Singapore (see Table 5.2).

For further improvement of the selection method, a graphical picture analysis was considered. This analysis could include a face recognition to filter out photos containing people. Furthermore, this photo analysis could help to discover misplaced geotags by comparing the pictures of different clusters. Such an analysis was out of the scope of this work.

Therefore, the implementation and evaluation of an improved picture selection algorithm based on a graphical picture analysis could be the subject of future work.

Bibliography

[1] Ricci, F., Rokach, L., Shapira, B. In: Introduction to Recommender Systems Handbook, Springer-Verlag New York Inc., New York, New York. (2010)

[2] Schafer, J.B., Konstan, J., Riedl, J.: Recommender systems in e-commerce. In: Proceedings of the 1st ACM conference on Electronic commerce, ACM, New York, New York. (1999)

[3] arcgisonline.com: Map data. http://server.arcgisonline.com/ArcGIS/rest/services/ESRI_Imagery_World_2D/MapServer/export Accessed: 17-02-2016.

[4] Wikitravel contributors: Wikitravel : Zurich. http://wikitravel.org/en/Zurich Accessed: 08-01-2016.

[5] MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics, University of California Press, Berkeley, California. (1967)

[6] Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, The AAAI Press, Menlo Park, California. (1996)

[7] Google: Map data. https://www.google.ch/maps Accessed: 20-02-2016.

[8] Flickr photo: UserId 24232779@N00; PhotoId 2164675404; Not available any more.

[9] Flickr photo: UserId 24232779@N00; PhotoId 2163861137; Not available any more.

[10] Vaidotas Miseikis: Evening View at the ETH. https://www.flickr.com/photos/v4idas/6343739852 Accessed: 17-02-2016.

[11] Martin Cauchon: sainte chapelle. https://www.flickr.com/photos/lecabri/9262479190 Accessed: 17-02-2016.

[12] Jurek: Notre Dame. https://www.flickr.com/photos/49214499@N06/9443880057 Accessed: 17-02-2016.

[13] Conor MacNeill: Sainte-Chapelle. https://www.flickr.com/photos/thefella/6193101423 Accessed: 17-02-2016.

[14] Erlend Robaye: Sainte Chapelle, Paris, France :: HDR :: Fisheye. https://www.flickr.com/photos/erroba/2988962403 Accessed: 17-02-2016.

[15] Artie Ng: The Stained Glass Windows of Sainte-Chapelle, Paris, France :: HDR. https://www.flickr.com/photos/artiephotography/11261076283 Accessed: 17-02-2016.

[16] Conor MacNeill: Lower Chapel. https://www.flickr.com/photos/thefella/6340526351 Accessed: 17-02-2016.

[17] Robert Cross: Let There Be Light. https://www.flickr.com/photos/53400673@N08/10624713825 Accessed: 17-02-2016.

[18] Els: Sainte-Chapelle, Ile de la Cite, Paris. https://www.flickr.com/photos/elsa11/16204773713 Accessed: 17-02-2016.

[19] Wikitravel contributors: Wikitravel : Singapore. http://wikitravel.org/en/Singapore Accessed: 08-01-2016.

[20] Wikitravel contributors: Wikitravel : Singapore, Riverside. http://wikitravel.org/en/Singapore/Riverside Accessed: 08-01-2016.

[21] Wikitravel contributors: Wikitravel : Singapore, Orchard. http://wikitravel.org/en/Singapore/Orchard Accessed: 08-01-2016.

[22] Wikitravel contributors: Wikitravel : Singapore, Marina Bay. http://wikitravel.org/en/Singapore/Marina_Bay Accessed: 08-01-2016.

[23] Wikitravel contributors: Wikitravel : Singapore, Bugis. http://wikitravel.org/en/Singapore/Bugis Accessed: 08-01-2016.

[24] Wikitravel contributors: Wikitravel : Singapore, Chinatown. http://wikitravel.org/en/Singapore/Chinatown Accessed: 08-01-2016.

[25] Wikitravel contributors: Wikitravel : Singapore, Little India. http://wikitravel.org/en/Singapore/Little_India Accessed: 08-01-2016.

[26] Wikitravel contributors: Wikitravel : Singapore, Balestier. http://wikitravel.org/en/Singapore/Balestier Accessed: 08-01-2016.

[27] Wikitravel contributors: Wikitravel : Singapore, North. http://wikitravel.org/en/Singapore/North Accessed: 08-01-2016.

[28] Wikitravel contributors: Wikitravel : Singapore, West. http://wikitravel.org/en/Singapore/West Accessed: 08-01-2016.

[29] Wikitravel contributors: Wikitravel : Singapore, Jurong. http://wikitravel.org/en/Singapore/Jurong Accessed: 08-01-2016.

[30] Wikitravel contributors: Wikitravel : Singapore, North-East. http://wikitravel.org/en/Singapore/North_East Accessed: 08-01-2016.

[31] Wikitravel contributors: Wikitravel : Singapore, Tampines. http://wikitravel.org/en/Singapore/Tampines Accessed: 08-01-2016.

[32] Wikitravel contributors: Wikitravel : Singapore, East-Coast. http://wikitravel.org/en/Singapore/East_Coast Accessed: 08-01-2016.

[33] Wikitravel contributors: Wikitravel : Singapore, Sentosa. http://wikitravel.org/en/Singapore/Sentosa Accessed: 08-01-2016.

[34] Wikitravel contributors: Wikitravel : Singapore, North-West. http://wikitravel.org/en/Singapore/North_West Accessed: 08-01-2016.

[35] Wikitravel contributors: Wikitravel : Paris. http://wikitravel.org/en/Paris Accessed: 08-01-2016.

[36] Wikitravel contributors: Wikitravel : Paris, 1st Arrondissement. http://wikitravel.org/en/Paris/1st_arrondissement Accessed: 08-01-2016.

[37] Wikitravel contributors: Wikitravel : Paris, 2nd Arrondissement. http://wikitravel.org/en/Paris/2nd_arrondissement Accessed: 08-01-2016.

[38] Wikitravel contributors: Wikitravel : Paris, 3rd Arrondissement. http://wikitravel.org/en/Paris/3rd_arrondissement Accessed: 08-01-2016.

[39] Wikitravel contributors: Wikitravel : Paris, 4th Arrondissement. http://wikitravel.org/en/Paris/4th_arrondissement Accessed: 08-01-2016.

[40] Wikitravel contributors: Wikitravel : Paris, 5th Arrondissement. http://wikitravel.org/en/Paris/5th_arrondissement Accessed: 08-01-2016.

[41] Wikitravel contributors: Wikitravel : Paris, 6th Arrondissement. http://wikitravel.org/en/Paris/6th_arrondissement Accessed: 08-01-2016.

[42] Wikitravel contributors: Wikitravel : Paris, th Arrondissement. http://wikitravel.org/en/Paris/7th_arrondissement Accessed: 08-01-2016.

[43] Wikitravel contributors: Wikitravel : Paris, th Arrondissement. http://wikitravel.org/en/Paris/8th_arrondissement Accessed: 08-01-2016.

[44] Wikitravel contributors: Wikitravel : Paris, 9th Arrondissement. http://wikitravel.org/en/Paris/9th_arrondissement Accessed: 08-01-2016.

[45] Wikitravel contributors: Wikitravel : Paris, 10th Arrondissement. http://wikitravel.org/en/Paris/10th_arrondissement Accessed: 08-01-2016.

[46] Wikitravel contributors: Wikitravel : Paris, 11th Arrondissement. http://wikitravel.org/en/Paris/11th_arrondissement Accessed: 08-01-2016.

[47] Wikitravel contributors: Wikitravel : Paris, 12th Arrondissement. http://wikitravel.org/en/Paris/12th_arrondissement Accessed: 08-01-2016.

[48] Wikitravel contributors: Wikitravel : Paris, 13th Arrondissement. http://wikitravel.org/en/Paris/13th_arrondissement Accessed: 08-01-2016.

[49] Wikitravel contributors: Wikitravel : Paris, 14th Arrondissement. http://wikitravel.org/en/Paris/14th_arrondissement Accessed: 08-01-2016.

[50] Wikitravel contributors: Wikitravel : Paris, 15th Arrondissement. http://wikitravel.org/en/Paris/15th_arrondissement Accessed: 08-01-2016.

[51] Wikitravel contributors: Wikitravel : Paris, 16th Arrondissement. http://wikitravel.org/en/Paris/16th_arrondissement Accessed: 08-01-2016.

[52] Wikitravel contributors: Wikitravel : Paris, 17th Arrondissement. http://wikitravel.org/en/Paris/17th_arrondissement Accessed: 08-01-2016.

[53] Wikitravel contributors: Wikitravel : Paris, 18th Arrondissement. http://wikitravel.org/en/Paris/18th_arrondissement Accessed: 08-01-2016.

[54] Wikitravel contributors: Wikitravel : Paris, 19th Arrondissement. http://wikitravel.org/en/Paris/19th_arrondissement Accessed: 08-01-2016.

[55] Wikitravel contributors: Wikitravel : Paris, 20th Arrondissement. http://wikitravel.org/en/Paris/20th_arrondissement Accessed: 08-01-2016.

[56] Wikitravel contributors: Wikitravel : Paris, La Defense. http://wikitravel.org/en/Paris/La_Defense Accessed: 08-01-2016.

[57] Wikitravel contributors: Wikitravel : Seattle. http://wikitravel.org/en/Seattle Accessed: 08-01-2016.

[58] Wikitravel contributors: Wikitravel : Seattle, Downtown. http://wikitravel.org/en/Seattle/Downtown Accessed: 08-01-2016.

[59] Wikitravel contributors: Wikitravel : Seattle, Pioneer Sqare - International District. http://wikitravel.org/en/Seattle/Pioneer_Square-International_District Accessed: 08-01-2016.

[60] Wikitravel contributors: Wikitravel : Seattle, Queen Anne - South Lake Union. http://wikitravel.org/en/Seattle/Queen_Anne-South_Lake_Union Accessed: 08-01-2016.

[61] Wikitravel contributors: Wikitravel : Seattle, Capitol Hill - Central District. http://wikitravel.org/en/Seattle/Capitol_Hill-Central_District Accessed: 08-01-2016.

[62] Wikitravel contributors: Wikitravel : Seattle, Ballard. http://wikitravel.org/en/Seattle/Ballard Accessed: 08-01-2016.

[63] Wikitravel contributors: Wikitravel : Seattle, Fremont. http://wikitravel.org/en/Seattle/Fremont Accessed: 08-01-2016.

[64] Wikitravel contributors: Wikitravel : Seattle University District. http://wikitravel.org/en/Seattle/University_District Accessed: 08-01-2016.

[65] Wikitravel contributors: Wikitravel : Seattle, North. http://wikitravel.org/en/Seattle/North Accessed: 08-01-2016.

[66] Wikitravel contributors: Wikitravel : Seattle, Sodo - Georgetown. http://wikitravel.org/en/Seattle/Sodo-Georgetown Accessed: 08-01-2016.

[67] Wikitravel contributors: Wikitravel : Seattle, South. http://wikitravel.org/en/Seattle/South Accessed: 08-01-2016.

[68] Wikitravel contributors: Wikitravel : Seattle, West. http://wikitravel.org/en/Seattle/West Accessed: 08-01-2016.