Predicting Photograph Seasons using Geo-tagged Images

Shesha B. Sreenivasamurthy
Univ. of California, Santa Cruz
ssreeniv@ucsc.edu

Shayna Frank
Univ. of California, Santa Cruz
smfrank@ucsc.edu

December 15, 2014

Abstract

Tens of thousands of pictures are taken at different locations throughout the year. People often visit places and take pictures to remember their visits. We believe that the seasonal travel patterns of tourists to specific locations will create a correlation between a location and the season of the images taken in that location. For example, fewer people visit Bear Valley during the summer than during the winter as it is a popular Ski Resort. Therefore, we believe we will find more pictures of Bear Valley taken during winter when compared to any other seasons.

Today, most of the photographs have geo-location (Latitude and Longitude) and the time when the picture was taken included in their their metadata. We use this metadata to predict the season in which a picture was taken when given the location in which it was taken. Correlations between locations and seasons found using this metadata could also potentially be used to predict the best time of year to visit particular locations or how many people may visit a tourist destination in the next season allowing business establishments to prepare appropriately.

In our work, using a dataset comprised of photograph metadata, we focused on approximately 1.1 million photographs taken in California. Using variations of the nearest neighbor algorithm, we were able to predict the season of a photograph with a maximum correctness of 80.9% with a sufficiently large training set. We experimented with both weighted K-Nearest Neighbor (K-NN) and Fixed Radius Nearest Neighbor (FRNN) using no-weight, inverse, logarithmic, and gaussian weight calculators. Using the K-NN model, we found that logarithmic weighted K-NN performed the best at 79.55%. Using the Fixed Radius NN model, we found that the gaussian weighted model using $\sigma = 10^{-4}$ peformed the best at 80.9%.

1 Introduction

With the advent of digital photography millions of pictures are taken and uploaded to social media sites including (but not limited to) Facebook, Flickr, Instagram and Picasa. The digital cameras and iPhones today are equipped with GPS systems that enable GPS co-ordinates, such as latitude and longitude of the place where the photograph is taken, to be linked to each photograph taken. By getting the GPS time, the exact time at which photograph was shot is also recorded. This information is stored

in the photograph's EXIF (exchangeable image file format) headers which can be extracted by many freely available tools. When people upload photographs to social media, the headers are extracted and stored as metadata in addition to other information such as photo-id, user-id, upload-time etc. This metadata is exposed by these social media sites via their published APIs.

We believe that the number of photographs taken in a particular vicinity are representative of the tourism patterns for those locations. As many locations have seasonal tourism patterns, we believe the location in which a picture was taken will be strongly correlated with the season of the year in which it was taken. With sufficiently large dataset, machine learning algorithms can utilize this relation to predict the season in which an image was taken given its location. The applications for this information are not limited to predicting the season in which a photograph was taken. Given the distribution of photographs taken during the year, the best time of year to visit certain location could be predicted. Additionally, the number of unique users associated with a location's photographs, when evaluated by season, could be representative of the number of people visiting that place during each season. This information can be used by businesses in that location to make the necessary preparations for the number of people visiting that place each season. There are a lot of such use cases that become possible with the availability of such metadata information.

When we were searching for a project (after the first unsuccessful choice) we read that Flickr had released 100 million pictures for the research community. Some research revealed rich Flickr APIs providing access to a plethora of information in the form of photograph metadata. One piece of information that caught our attention was that of geo-location and we came up with the idea of predicting a pictures season based upon where the picture was taken. We decided to employ the season(date) information of photographs taken in close proximity to a given photograph, suggesting a model that aligns nicely with nearest neighbor technique.

With guidance from Professor Helmbold, we decided to evaluate the efficacy of two learning algorithms: K-Nearest Neighbors and Fixed-Radius Nearest Neighbors. We evaluate each algorithm's performance with a varying number of training samples and with different weighting techniques such as no-weight, inverse distance weight, log distance weight and gaussian weights. For the gaussian weighting technique, we explore the accuracy of our classification with varying σ values. For the K-NN algorithm, we assess the performance of the algorithm when employing a varying number of nearest neighbors. For the FRNN algorithm, we survey the algorithm's performance with varying radius sizes.

The remaining of the document is organized as follows: background information is provided in section 2 and our system is explained in section 3. Results and experiments are given in section 4 and conclusion in section 5.

2 Background

We drew inspiration from the work done by Zhou, Bolei, et al. [1] where the identity of the city is determined by inspecting the contents of the geo-tagged photograph. Seven high level attributes: vertical buildings, water coverage, green space coverage, type of architecture, transportation, athletic and social activity, is used to describe the city. However, in this work we just access the metadata of the photograph rather than the data of the photograph to predict the season in which the photograph was taken. Gallagher, Andrew, et al. [11] have shown significant improvement in geo-location

inference of a photograph when geo-location information in the metadata is used in addition to the visual contents of the photograph.

The ICSI (International Computer Science Institute) used the same Flickr dataset for audio and video recognition techniques that can reliably identify the geographic locations of the photograph or video. The dataset is also used to segment the video files in order to treat similar sounds as keywords for improving search accuracy [7].

The Flickr dataset is also used by Weyand, Tobias et al [8]. In this project, the rich Flickr API is used to gain access to the same metadata of a set of photographs. The dataset was used for the evaluation of large-scale landmark discovery algorithms. Wang, Josiah K., et al. [9] used the same dataset to evaluate their granularity aware grouping in order to enable the fair evaluation of a a system meant for precise annotations (poodle vs dog) against a system that annotates with coarser granularity, as both annotations are correct.

In the rest of this section we briefly describe the algorithms that we have used in our project to aid in the understanding of our results.

2.1 K-Nearest Neighbors

In this algorithm the data in the training set is remembered. In its most basic form, $K = 1$ and each test point is predicted to have the same labels as its nearest training point (nearest neighbor). This can be semi-formally stated as:

For a given point $q \; \varepsilon \; T$**, find set of points** $p_i \; \varepsilon \; N \mid i \leq K, \; N \subset S, \; distance(q, p_i) =$ **least over all points in** S, where:
T = Test Set
S = Training Set
N = Nearest Neighbor Set
K = K-Nearest Neighbors

In order to improve the accuracy, in practice K is set to a value greater than 1 such that K nearest points are considered before predicting and the label of majority of the K nearest points is predicted to be the label of the new test point. Weighting techniques can also be applied to each of the neighbors in order to improve the model's performance. This use of weighting techniques enables an extension of the model designating a greater amount of influence to the labels of samples closer to the test point than those of more distant samples when determining the classification of a test point.

There are different techniques to associate weights with training samples. With inverse distance and log distance weighting, closer neighbors have a greater influence, but there are no parameters available to further tune which samples will exert a higher influence on the predictions. However, if we consider a gaussian around each point, variation of the standard deviation can adjust the extent to which more immediate neighbors influence the ultimate prediction. With a small value for σ, the peak of the gaussian is high and the slope is steep. This translates into near points having significantly higher influence than those slightly farther away. As σ increases, the peak is less pronounced and the discrepancy between weights given to very close and slightly more distant points decreases leading toward more uniform influence for all neighbors.

Nearest neighbor is considered as an 'instance based learning' model, as only local points are considered and 'lazy' learning algorithm as actual prediction is deferred until classification. This is appropriately named as 'Lazy Instance Based - K' in Weka [5]. Various improvements to the algorithm have been suggested in the research community. The MFS (Multi Feature Subsets) algorithm combines multiple NN classifiers by dividing features into a random num-

ber of subset of features, each having an equal number of features [6]. The number of features is a hyper-parameter that is determined by cross validation. MFS significantly improves performance of NN and is also more robust to corruption by irrelevant features compared than K-NN.

2.2 Fixed-Radius Clustering

The Fixed-Radius Nearest Neighbor algorithm is similar to the K-NN algorithm in that the data in the training set is remembered and the neighbors of a given test point are polled in order to determine the point's classification [10]. It differs in that, in addition to the parameter K, a radius parameter R determines the area from which neighbors are selected. In FRNN, the K parameter represents the *maximum* number of nearest neighbors to be selected from within the constraints of the radius R. The algorithm is frequently run with either a very large (or infinity) value for K in order to allow R to solely determine the selection of relevant neighbors. If K is less than the number of neighbors contained in the radius, only the K nearest neighbors are considered.

Once the neighbors are chosen using the two parameters, the ultimate classification is determined by the same method employed by the K-NN algorithm. The same weighting techniques can be utilized to vary the importance levels of points with respect to their distance from the test point.

3 System

Given a photo and the location in which it was taken, our system is designed to predict the season in which the photo was taken. We have evaluated our system using two instance based algorithms: 1. K-Nearest Neighbor method and 2. Fixed-Radius Clustering.

3.1 Dataset Pre-processing

For our project, a large number of geo-tagged images with time stamps are necessary. Yahoo Labs announced in the middle of this year that they released 100 Million Flickr images, including their metadata, for use in research [2]. This is one of the largest multimedia dataset ever released to public [7]. We were able to get access to the dataset after a formal request detailing our proposed useage. The metadata contains 23 fields of which we employed *photo-id, user-id, date-taken, date-uploaded, latitude, longitude* and *accuracy*. The first three elements are relatively self explanatory. The accuracy attribute refers to the precision of the location information associated with the photograph. It is an integer value that ranges from 1-16 with 1 being least amount of accuracy and 16 being the greatest amount of accuracy.

Flickr has following convention for accuracy: World level is 1, Country is $\sim$3, Region $\sim$6, City $\sim$11, Street $\sim$16. It defaults to 16 if accuracy is not specified. For our experimentation, we filtered the photographs to only consider those with an accuracy value between 11 and 15. We did not included photographs with an accuracy of 16, though the group includes the most accurate samples, as it also the default value for those photographs to which the user has not specified the accuracy and therefore introduces a large amount of noise when included. This decision is bolstered by a study by Hauff, Claudia [10] that shows accuracy of the geotag information is highly dependent on the popularity of the location. Therefore, un-popular images gets tagged with default value of 16 that introduces noise in our dataset.

The Flickr dataset contains a total of 100 million samples from all over the world. For our work we narrowed our focus to only those pictures taken in California (figure 1) by considering the pictures that fall within the following

Figure 1: California Latitude Longitude Co-Ordinates [4]

$<latitude, longitude>$ range:
$(42.0°, -125.0°)$ and $(32.0°, -120.0°)$
$(36.0°, -114.0°)$ and $(36.0°, -120.0°)$
$(36.0°, -116.0°)$ and $(38.0°, -120.0°)$ and
$(38.0°, -119.0°)$ and $(39.0°, -120.0°)$.

After the above pre-processing procedure, we ended up with approximately 1.1 million samples.

Once we extracted the photographs taken in California, we had to convert latitude and longitude to cartesian co-ordinates so that we could approximate distances using the Euclidean distance equation. Latitude and Longitude co-ordinates for spherical earth are converted to three dimensional planar co-ordinates, increasing the number of features in our dataset from 2 to 3.

The conversion details are as follows:

$$x = R \times cos(latitude) \times cos(longitude)$$
$$y = R \times cos(latitude) \times sin(longitude)$$
$$z = R \times sin(latitude)$$

where:

$R =$ Radius of the earth (6371 KMs)

x	y	z	season
-2451.9303	-4733.2923	3489.0718	SU
-2500.4700	-4649.6220	3566.2733	FA
-2474.2460	-4294.0718	4003.5851	WI
-2409.7223	-4498.5753	3813.8823	FA
-2425.9066,	-4733.3593	3507.1251	SP
-2601.9983	-4495.9003	3688.6481	SP
-2678.2435	-4427.81068	3716.3348	SU
-2485.2648	-4687.1492	3527.5673	FA
-2697.4147	-4250.8025	3904.3914	FA
-2673.6398	-4228.6388	3944.6045	SP

Table 1: Dataset after pre-processing

The distance between any two points is then calculated using Euclidean distance formula:

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

Each picture is then labeled as Winter (WI), Spring (SP), Summer (SU) and Fall (FA) based on the date on which the photograph was taken. We use official seasonal dates for classification.

Winter (WI): December 21^{st} - March 20^{th}
Spring (SP): March 21^{st} - June 20^{th}
Summer (SU): June 21^{st} - September 22^{nd}
Fall (FA): September 23^{rd} - December 20^{th}.

The dataset resulting from our various forms of pre-processing is sampled in table 1. The co-ordinates are given in radians.

3.2 K-Nearest Neighbors

Each of our training samples contains latitude and longitude as features and season as their labels. We performed the same pre-processing on the test data as we did on the training data. Then, using K-Nearest Neighbors, we predict the season (label) for each new test sample using different weighting techniques.

When considering K-Nearest Neighbor algorithm, there are different parameters that can be tuned in order to achieve accurate predictions. The value of K in K-NN is the hyper-parameter determining the number of neighbors polled, but the number of training samples used to create the prediction model can also be varied and has a significant impact on model performance. Additionally, all neighbors need not be considered as equal. It makes sense to allocate more weight to the points that are closer to the test point when compared to others so that training points farther from the test point have less influence on the prediction. Below is a brief description of the methods we used to assign weights to the neighboring training points based on their distance from the test point.

1. **No-Weight:** Each point is assigned a constant weight of 1, resulting in all points being treated equally by not associating any weights or by associating constant weights to them.

2. **Inverse Weight:** Inverse of the distance is used for the weight. Closer points have small distances and the inverse associates a heavier weight when compared to the points farther away, which will end up having smaller weight.

3. **Logarithmic Weight:** Associating a negative logarithm to the distance will have a similar effect to that of the Inverse Weight approach but there is a slight bias toward closer points due to the shape of the curve.

4. **Gaussian:** A weight is associated to the training point based on the value of the point on a gaussian distribution with a mean of zero. By varying standard deviation we can modify the discrepancy between weights assigned to closer and more distant training points, but generally, distances closer to zero (the mean used for the curve) result in higher weights as they are closer to the peak of the curve. Distances farther from the mean fall on the tails of the distribution and therefore are assigned lower weights.

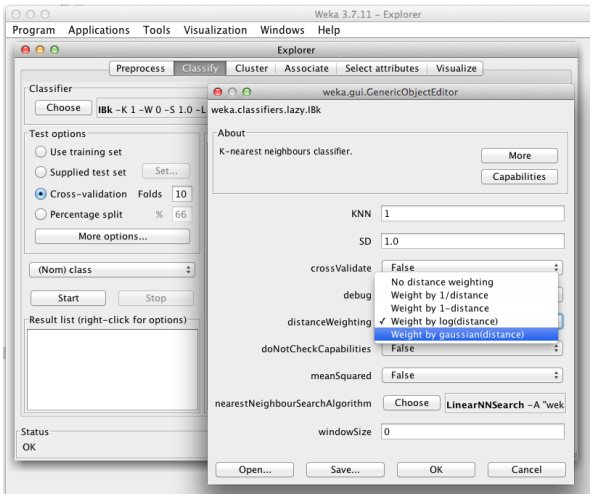


Figure 2: Improvements to Weka [5]

To evaluate different K-NN methods, we used *Weka*, an open source data mining software in Java [5]. Weka supports No-Weights(constant) and Inverse Weights for K-NN algorithm. Therefore, we extended the Weka tool to support Logarithmic and Gaussian weights. We modified Weka to accept standard deviation (σ) as input for experimenting with different values for the σ hyper-parameter. For calculating logarithmic weight we used natural logarithm ($log_e(x)$) and for gaussian we used the normal distribution formula: $f(x,\mu,\sigma) = \frac{1}{\sqrt{2\sigma^2}}\, e^{\frac{-(x-\mu)^2}{2\sigma^2}}$, with μ (mean) set to zero and experimenting with different σ (standard deviation) values as hyper-parameters. Our improvements to Weka is shown in figure 2. A detailed explanation of the experiments that we conducted along with the results are summarized in the next section.

3.3 Fixed-Radius Nearest Neighbors

As far as we could find, there was no support for Fixed Radius Nearest Neighbor evaluation in *Weka* so we set about to implement the algorithm on our own. After unsuccessful partial implementations using a C++ library called *ANN* (`http://www.cs.umd.edu/~mount/ANN/`) and *SciKit Learn* (`http://scikit-learn.org/`) for Python, we resorted to using an existing Java implementation for KD trees as a base and writing our own implementation of the FRNN algorithm. Both the *ANN* and *SciKit Learn* implementations of the algorithm ended up having deeply embedded calculation exceptions when the radius parameter was small (less than .001) and we wanted to test with radii as small as $1e^{-9}$, so neither library was able to handle our requirements. Additionally, writing our own implementation of the algorithm allowed for a greater amount of customization to vary the algorithm's parameters.

The parameters that can be adjusted for the FRNN algorithm are the radius parameter R and the other is the method in which points that are unable to be classified in the normal manner are handled H. One inevitable complication with the FRNN algorithm is that, depending on the chosen radius and the density of points in the sample area, there is the possibility of having test points with no neighbors. In order to evaluate the best way to handle such points we experimented with the following three methods:

1. **Strict Handler:** The *Strict Handler* did not attempt to classify points with no relevant neighbors. The points were simply labeled as unclassified and considered to be incorrect predictions.

2. **Random Handler:** In the event of an unclassifiable point, the *Random Handler* chose the photograph's season to prediction randomly from the four possible options.

3. **Nearest Handler:** The *Nearest Handler* predicted the season associated with the single closest neighbor, regardless of the fact that the neighbor was not located in the specified radius.

In the cases where there were neighbors within the specified radius, the same weighting techniques were employed for prediction as were described for the K-NN method.

4 Experiments and Results

The dataset that contains metadata of 100 million photographs that we got from Flickr was split into 10 equal sets of 10 million. We ran our pre-processing algorithm on each set to extract the metadata for California and filter out data that are not between the accuracy range of 11 and 15. The first 9 sets were used for constructing training sets of various sizes while the last set was used for creating validation and test-sets. Usually 20% to 30% of training samples are used for testing. However, when the training dataset is in the order of millions, running the nearest neighbor algorithm with a large test-set takes quite a bit of time, making running many experiments to evaluate different hyper-parameters impractical.

Therefore, we gathered all our test results using 2K test samples. To measure the impact of using less testing samples we conducted a simple experiment. We took 120K samples of which 80K were used as training set. We measured the correctness difference versus time saved when we used 40K and 2K test samples. From figure 3, we can see that for a very small decrease in correctness in the order of 1-5% (which is not even visible in the figure) we save experiment time of approximately 2000%.

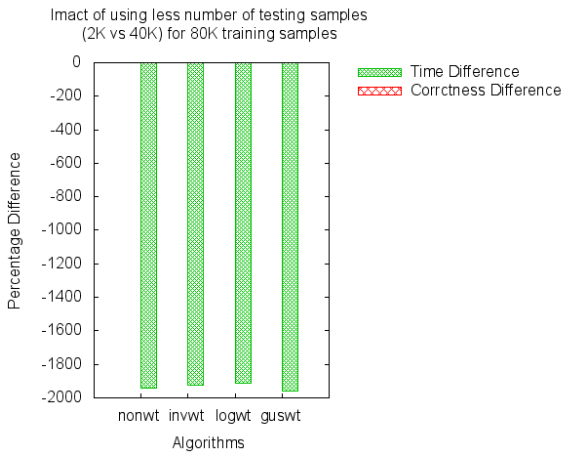


Figure 3: Impact of using less number of testing samples (2K vs 40K) for 80K training samples

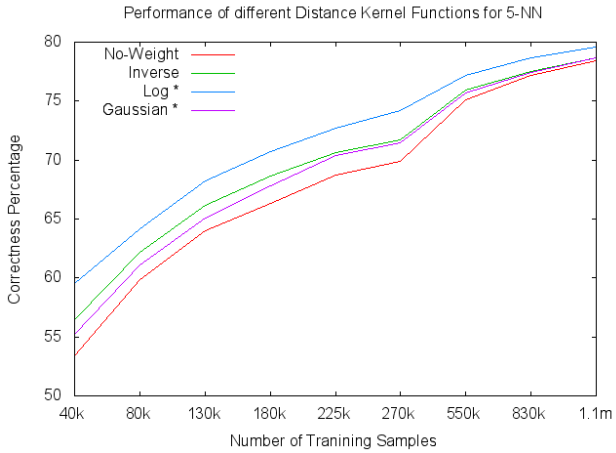


Figure 4: Performance of different Distance Kernel Functions for 5-NN

Therefore, we though that it is a worthwhile tradeoff and conducted our experiments with 2K test samples. The negative sign indicates decrease in correctness and decrease in time to complete each experiment.

4.1 K-Nearest Neighbors

4.1.1 Number of Training Samples

We started off by measuring the percentage of test samples that were correctly predicted (correctness) for different number of training samples and for different weighted K-NN weighting techniques. For this experiment a K value of 5 was chosen. The results are as shown in figure 4. The correctness increases as the number of training samples increases for all weighting techniques. However, all techniques starts to converge towards 80%. The maximum correctness of 79.55% was measured with logarithmic distance weighting. We can infer that, when the number of training samples are low, logarithmic weighting performs the best and as the number of training samples increases, the choice of weighting technique becomes insignificant for our dataset. For each class WI, SP, SU and FA, accuracy of correct predictions and log-loss of mis-predictions were measured.

$$accuracy = \frac{Number\ of\ TP\ for\ class_i}{Total\ samples\ of\ class_i}$$

$$log\text{-}loss = \frac{-\Sigma\ log_e(P(Actual\ Class_i))}{Total\ mis\text{-}predictions\ of\ class_i}$$

Figures 5 and 6 gives accuracy and log-loss for all four seasons for different training samples for log-weighted K-NN. We are just providing results of log-weighted K-NN as it performed the best in our experiments. We can infer that, as accuracy increases log-loss decreases. The increase in accuracy and decrease in log-loss is almost linear with respect to number of training samples. Therefore, more the training samples, better will be our prediction.

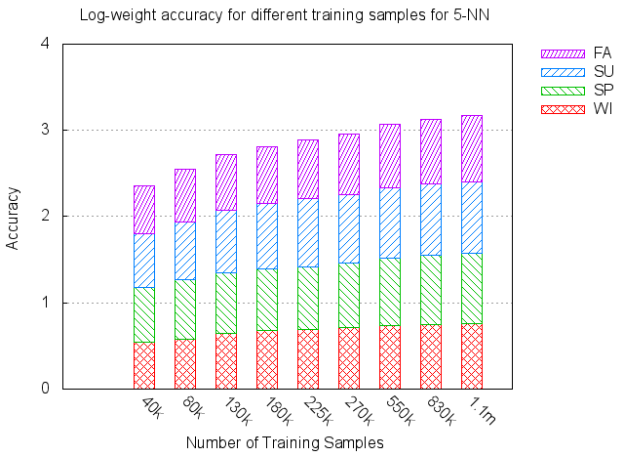


Figure 5: Log-weight accuracy for different training samples for 5-NN

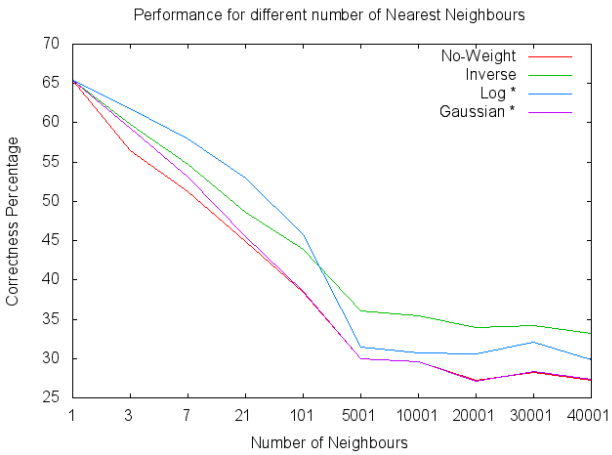


Figure 7: Performance of different number of nearest neighbors with 40K training and 2K test samples

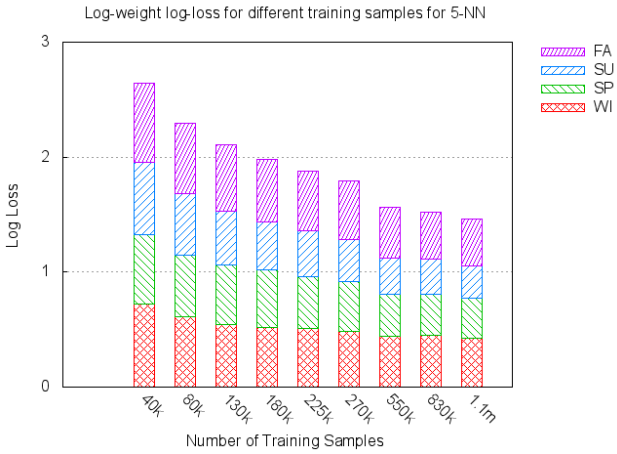


Figure 6: Log-weight log-loss for different training samples for 5-NN

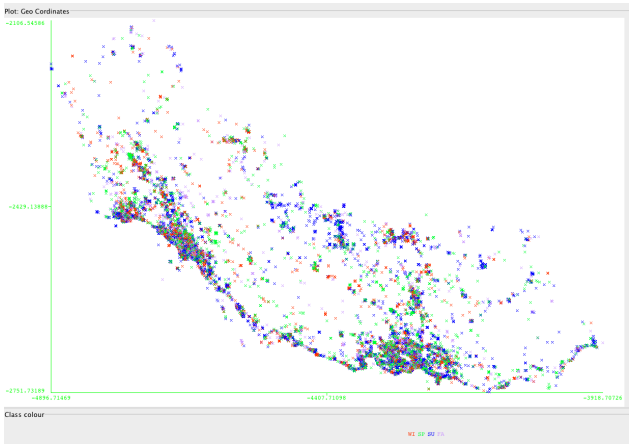


Figure 8: 40K Training Sample Distribution from Weka

4.1.2 Number of Nearest Neighbors

The next hyper-parameter that we experimented with is number of nearest neighbors. For this experiment we considered 40K training samples with 2K test samples. The argument for using less than 20% of training samples as test samples given previously holds good in this case too.

Performance of different number of nearest neighbors with 40K training and 2K test samples is as shown in figure 7. From the graph we can see that correctness decreases as the number of nearest neighbors increases and flattens out when N reaches number of training samples. We would expect an initial increase in correctness with the increase in number of nearest neighbors due to the decreased influence of possible noise, followed by a decrease in correctness due to the eventual inclusion of irrelevant training points. We believe that the reason for such a behavior is the way our data is distributed.

Figure 8 shows the distribution of our dataset. In our dataset, SP and SU class samples are clustered more when compared to WI and FA class samples. Therefore, when predicting SP

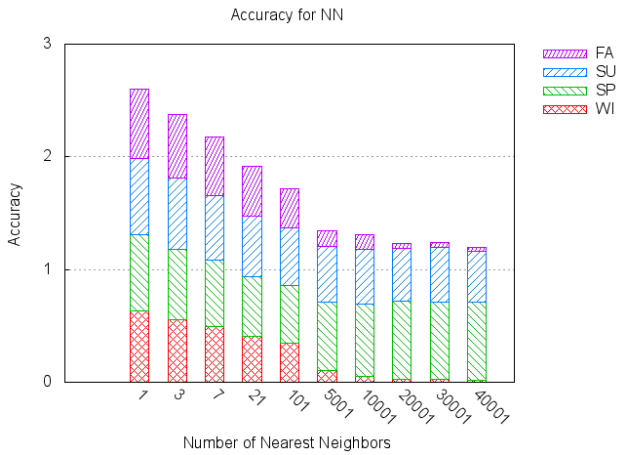


Figure 9: Accuracy for NN

σ	40K	80k	130k	180k	225k
10^{-9}	1.0883	0.8158	0.7214	0.6475	0.6052
10^{-8}	1.0871	0.8158	0.7215	0.6473	0.6078
10^{-7}	1.0865	0.8065	0.6985	0.6297	0.5825
10^{-6}	1.0221	0.7413*	0.6592*	0.5731*	0.5457*
10^{-5}	0.9378	0.7553	0.6603	0.5799	0.5604
10^{-4}	0.9348*	0.7788	0.6715	0.5960	0.5557
10^{-3}	0.9508	0.7907	0.6842	0.6020	0.5605
10^{-2}	0.9471	0.7895	0.6824	0.6003	0.5606
10^{-1}	0.9471	0.7895	0.6825	0.6003	0.5606
1	0.9471	0.7895	0.6825	0.6003	0.5606

Table 2: Log-loss to Accuracy ratio for 3-NN

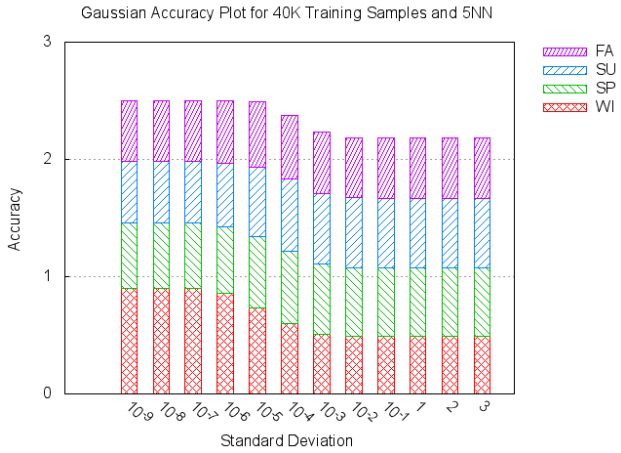
and SU class, increasing number of neighbors increases the accuracy with which we predict those classes. WI and FA points are relatively scattered. Therefore, a number of samples around WI and FA class belong to different class and K-NN mis-predicts. In case of weighted K-NN the penalty is high as non WI and FA neighbors too will have heavy weights and will drag down total correctness. This is evident from the accuracies shown in figures 9. We can infer that the accuracy of SP and SU class remains the same or slightly increases as we increase the number of nearest neighbors but the accuracy of WI and FA classes decreases sharply which contributes to total decrease in the correctness that we saw in figure 7. For a scattered dataset, increasing number of nearest neighbors will have a detrimental effect on the overall prediction.

4.1.3 Gaussians

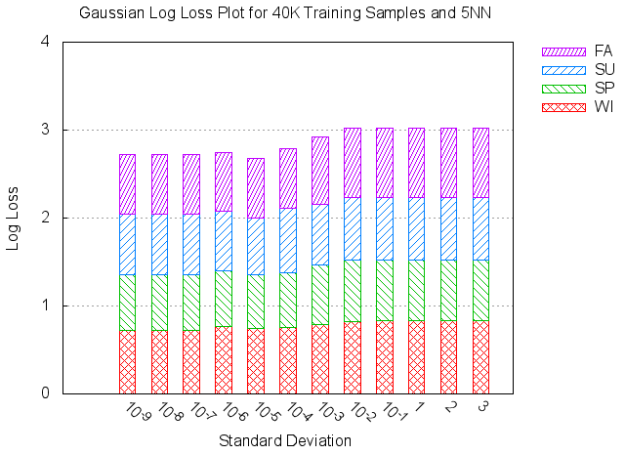
Next we experimented with gaussian weighted K-NN. We considered a gaussian with a mean of zero for each test sample point and experimented with various standard deviations (σ) as a hyper-parameter. The goal was to find the best value of σ that suits our dataset. We experimented with different number of nearest neighbors and different number of training samples.

When number of nearest neighbors increase, we can infer from all graphs in figure 10 that accuracy at low σ remains the same for different number of nearest neighbors. This is because using a low σ value, fewer neighbors have the majority of the influence on the ultimate prediction rendering the slightly more distant samples insignificant. As σ increases, more and more samples will get similar weights and more neighbors contribute to prediction. Accuracy of WI and FA has reduced as the samples for those classes are scattered and accuracy of SP and SU goes up or remains same as their samples are more clustered. Similarly, as σ increases, the log-loss increases for WI and FA classes only. Another thing to notice is, as σ decreases the samples very close to the test point will have heavy weight and the weight sharply decreases as we go little farther from mean. This means, a very low σ is almost equal to 1-NN. Therefore, we can see an increase in accuracy as we decrease σ for the reasons explained previously in section 4.1.2.

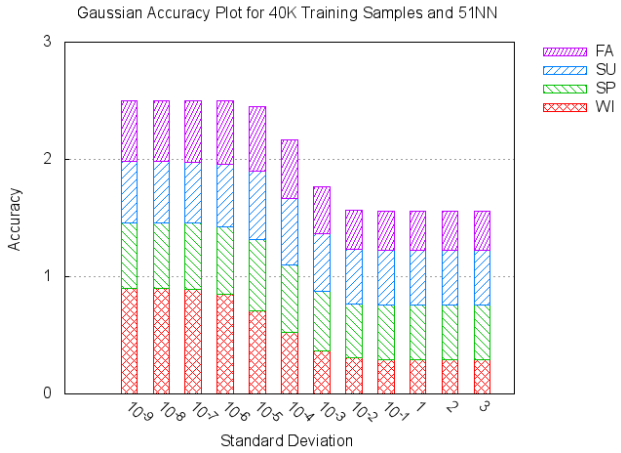
Comparing figured 10(a), 11(a) and figures 10(b), 11(b), as the number of training samples increase from 40K to 1.1M, accuracy increases and log-loss decreases. This observation is in sync with what we had observed in figure 4. To determine an appropriate σ for our dataset we conducted a total of 250 experiments



(a) 40K, 5-NN, Accuracy

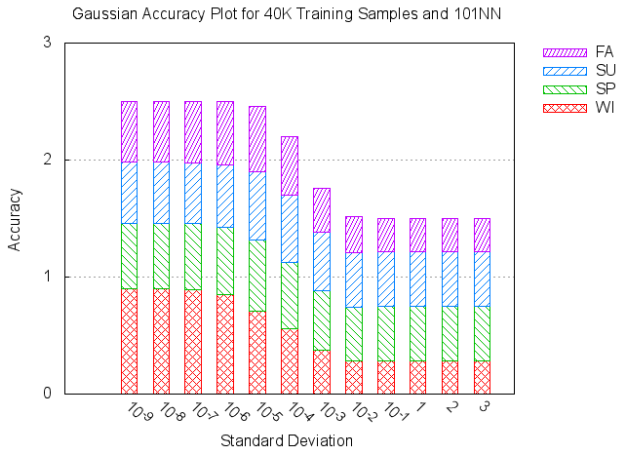


(b) 40K, 5-NN, Log-Loss

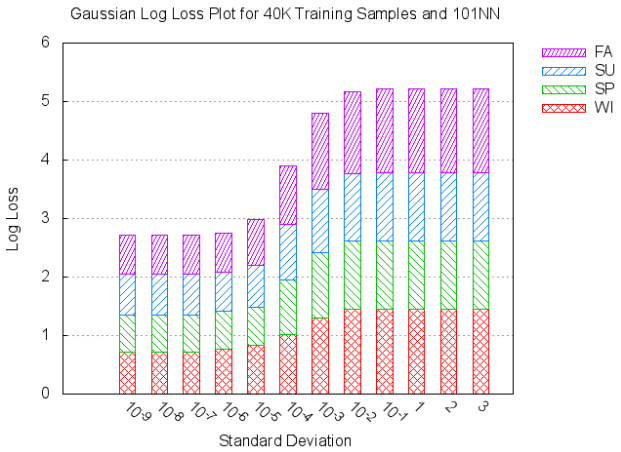


(c) 40K, 51-NN, Accuracy

(d) 40K, 51-NN, Log-Loss



(e) 40K, 101-NN, Accuracy



(f) 40K, 101-NN, Log-Loss

Figure 10: Accuracy and Log-Loss for different Gaussians for 40K samples

σ	40K	80k	130k	180k	225k
10^{-9}	1.0883	0.8158	0.7214	0.6475	0.6052
10^{-8}	1.0871	0.8158	0.7215	0.6473	0.6078
10^{-7}	1.0959	0.8085	0.7002*	0.6354*	0.5926*
10^{-6}	1.0752*	0.7985*	0.7121	0.6402	0.5992
10^{-5}	1.1775	0.9421	0.8229	0.7355	0.7017
10^{-4}	1.3064	1.0552	0.9109	0.8198	0.7462
10^{-3}	1.3821	1.0938	0.9310	0.8402	0.7649
10^{-2}	1.3830	1.0902	0.9296	0.8396	0.7638
10^{-1}	1.3831	1.0902	0.9296	0.8396	0.7638
1	1.3831	1.0902	0.9296	0.8396	0.7638

Table 3: Log-loss to Accuracy ratio for 5-NN

σ	40K	80k	130k	180k	225k
10^{-9}	1.0883	0.8158	0.7214	0.6475	0.6052
10^{-8}	1.0871*	0.8158	0.7215	0.6473	0.6078
10^{-7}	1.0981	0.8117*	0.7036*	0.6396*	0.5957*
10^{-6}	1.1684	0.8843	0.7868	0.7267	0.6843
10^{-5}	1.4888	1.2422	1.0674	0.9850	0.9520
10^{-4}	1.8462	1.5324	1.3224	1.2185	1.1287
10^{-3}	2.0186	1.6513	1.3864	1.2621	1.1862
10^{-2}	2.0285	1.6561	1.3867	1.2619	1.1854
10^{-1}	2.0285	1.6563	1.3867	1.2619	1.1854
1	2.0285	1.6563	1.3867	1.2619	1.1854

Table 4: Log-loss to Accuracy ratio for 11-NN

σ	40K	80k	130k	180k	225k
10^{-9}	1.0883*	0.8158	0.7214	0.6475	0.6052
10^{-8}	1.0884	0.8153	0.7215	0.6473	0.6078
10^{-7}	1.1032	0.8143*	0.7075*	0.6434*	0.5996*
10^{-6}	1.2135	0.9358	0.8552	0.8007	0.7791
10^{-5}	1.7971	1.5529	1.4083	1.3100	1.2844
10^{-4}	2.6833	2.2709	2.0640	1.9123	1.8412
10^{-3}	3.2503	2.6896	2.3500	2.1936	2.1057
10^{-2}	3.2871	2.7122	2.3631	2.2139	2.1271
10^{-1}	3.2873	2.7124	2.3633	2.2139	2.1272
1	3.2873	2.7124	2.3633	2.2139	2.1272

Table 5: Log-loss to Accuracy ratio for 51-NN

σ	40K	80k	130k	180k	225k
10^{-9}	1.0883*	0.8158	0.7214	0.6475	0.6052
10^{-8}	1.0884	0.8153	0.7215	0.6473	0.6078
10^{-7}	1.1033	0.8144*	0.7073*	0.6440*	0.6004*
10^{-6}	1.2099	0.9408	0.8650	0.8177	0.7898
10^{-5}	1.7749	1.5952	1.4825	1.4104	1.3876
10^{-4}	2.7303	2.4877	2.3443	2.2294	2.0987
10^{-3}	3.4157	3.0513	2.8376	2.6577	2.4493
10^{-2}	3.4751	3.1143	2.8584	2.6865	2.4682
10^{-1}	3.4755	3.1145	2.8587	2.6868	2.4684
1	3.4755	3.1145	2.8587	2.6868	2.4684

Table 6: Log-loss to Accuracy ratio for 101-NN

for different values of σ ($10^{-9} \leq \sigma \leq 1$), nearest neighbors ($3 \leq K \leq 101$) and training set ($40K \leq samples \leq 225K$). We computed log-loss to accuracy ratio in each of our experiments. The results are shown in tables 2-6. Asterix beside the values give the minimum log-loss to accuracy ratio and σ corresponding to the minimum ratio will be the most appropriate for our dataset. From the tables, we can infer that the most appropriate σ is 10^{-6} for 3-NN and is 10^{-7} for 5-NN, 11-NN, 51-NN and 101-NN. Therefore we can conclude that the most appropriate σ for our dataset is between 10^{-6} and 10^{-7}. Very low σ implies lesser neighbors and the reason why correctness and loss to accuracy ratio is better at lower σ for our dataset is reasoned out in section 4.1.2.

4.2 Fixed-Radius Nearest Neighbors

4.2.1 Number of Training Samples

For our Fixed-Radius Nearest Neighbors experimentation, we first evaluated the performance of the algorithm with a varying number of training points in order to find the optimal number of training points to use for the remainder of our experimentation. You can see in figures 12 and 13 that, with the FRNN algorithm, it is the gaussian weighting that results in the highest percentage of correct prediction, maxing out around 80% in all three cases. The inverse distance weighting also results in a high correctness percentage maxing out around 75% with the largest training set. The uniform weighting and log weighting techniques are less impressive with between 50 and 60% correctness at their best.

In addition to varying the size of the training set, we evaluated the effect of using each of the three methods for handling unclassified points. The correctness achieved by the three

(a) 1.1M, 5-NN, Accuracy

(b) 1.1M, 5-NN, Log-Loss

Figure 11: Accuracy and Log-Loss for different Gaussians for 1.1M samples

methods are within a few percentiles of each other, but as you can see in figure 12, the correctness achieved using the *nearest* or *random* methods provide the best results. The correctness spikes a bit more quickly when employing the *nearest* handler, but as the size of the training set increases, the correctness for both methods converge.

As would be expected, the performance of the *strict* handler is slightly worse in all cases. The most obvious distinction, however, is seen with smaller training sets, which is to be expected as there would be a greater percentage of test points with no true neighbors, all of which would be counted as incorrect predictions. The *random* and *nearest* handlers have a correctness of around ten percent better than the *strict* handler with smaller datasets, but as the datasets get larger, the difference is less apparent.

When weighing model performance and runtime, we decided that the subsequent experiments would be run using the training set with 225,000 samples. The results gathered from varying the size of the training set showed that with the FRNN algorithm, the performance when using training sets smaller than 225,000 drops off abruptly and the increase in performance with larger training sets increases at a relatively gradual pace resulting in at most an increase in correctness of about 5%. Additionally, the run time for the algorithm using the chosen training set size was not too unreasonable.

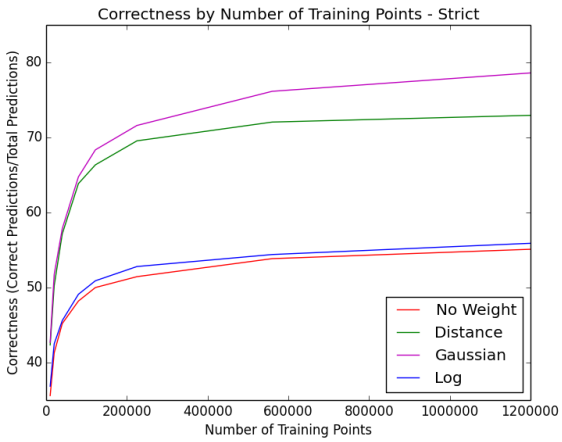
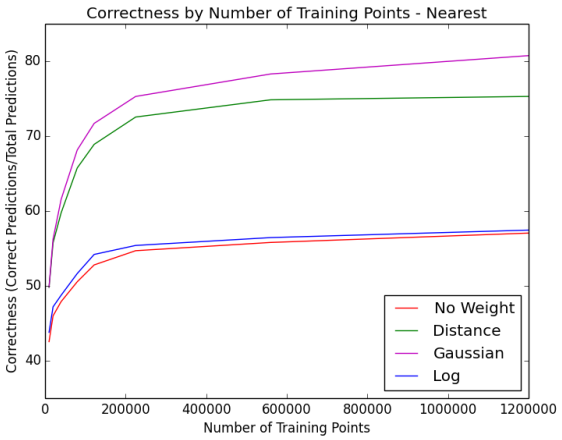


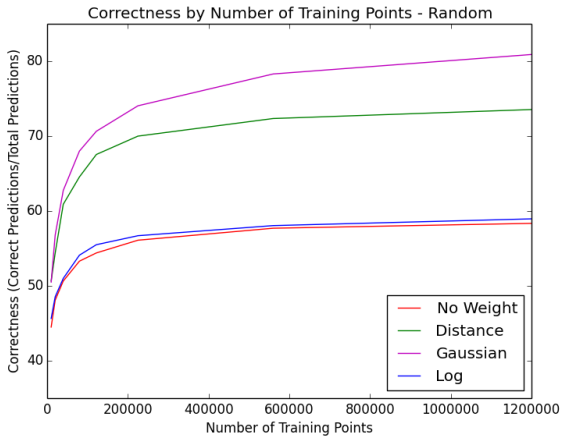
Figure 13: Correctness with varying training set sizes

4.2.2 Size of Radius

The primary parameter for the FRNN algorithm is the size of the radius used to select neighbors. When experimenting with a range of radii from $10^{-}9$ to 10, we had some interesting results. As seen in Figure 14, for the gaussian



(a) Nearest, Correctness



(b) Random, Correctness

Figure 12: Correctness with varying training set sizes

and distance weighting techniques, the percentage of correct predictions increases with an increase in the radius value, leveling off around 10^-1. In contrast, however, the logarithmic and uniform techniques decline quickly with radius values over about 10^-5. The behavior of the gaussian and distance curves can likely be attributed to the combination of an overall increase in input data, allowing for less noise disruption, while still attributing more influence to the closer points. However, since intuitively the logarithmic function would have similar properties, it would be interested future work to look into the reason for the stark contrast between the techniques.

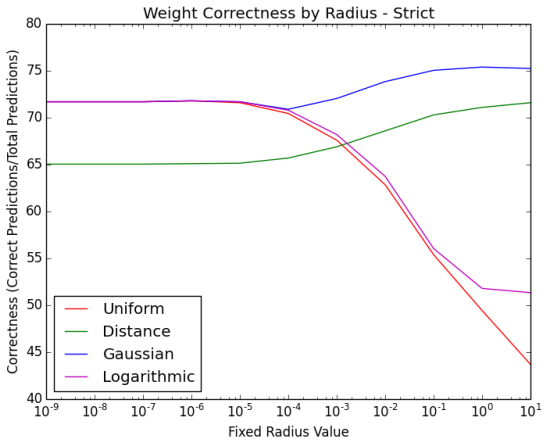


Figure 14: Correctness with varying radius (R) values

4.2.3 Gaussians

When experimenting with differing standard deviation (σ) values when employing the gaussian weighting technique, we found that, in general, regardless of the radius, σ-values of 10^-4 or smaller resulted in the highest correctness percentages. As this correlates to allocating almost all of the prediction influence to only the points closest to the test point, this reinforces our findings with the $K-NN$ algorithm where the fewer the neighbors, the higher the correctness percentage. Similarly, the overall accuracy dropped off at the same point as seen in Figure 15.

5 Conclusion

The goal of this study was to find the feasibility of using metadata information of geo-tagged photographs to predict the season in which a photograph was taken given its location and to evaluate variations of the nearest neighbor algorithm to fine the best models for this prediction.

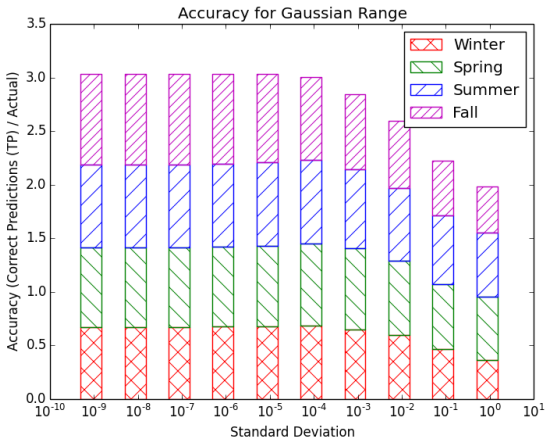


Figure 15: Accuracy with varying σ values

K-NN predicts with a maximum correctness of 79.55% and FRNN can achieve 80.9% correctness when trained using the approximately 1.1 million training samples. We experimented with different weighting algorithms like, no-weight, inverse weight, log weight and gaussian weights and found that, for K-NN, though logarithmic weight yields the best results, gaussian weighting with ($10^{-7} \leq \sigma \leq 10^{-6}$) performed well too. However for FRNN, gaussian weighting with small σ-values was the top performer. A maybe reason why log-weighted technique perform better in K-NN when compared to gaussian-weighted technique in FRNN is because, log-weight of distant points may have more negative impact when compared to gaussian-weight. In FRNN any points outside a given radius are not considered and therefore log-weights of distant points do not impact the prediction. However, more analysis and experimentation is required to actually prove it.

We used the Weka data mining tool in our course and were impressed by the power of the tool and the ease of its use. We started off using the unmodified version of the tool and later realized that it did not have all the functionality that we needed, like logarithmic weighted and gaussian weighted K-NN. We explored the idea of coding a plugin to Weka for the additional functionality we needed. However, as Weka was an open source tool, we decided to modify the tool itself. The code is very modular and classes are aptly named making it straightforward to modify. Throughout the course of the project we may have made close to 1000 runs with various configurations and the software did not crash a single time, which demonstrates its robustness.

For the Fixed-Radius Nearest Neighbor experimentation that was not supported by Weka, we implemented our own tool (code available upon request). We made the tool flexible enough to allowing for evaluation of the algorithm with varying parameters such as radius size (R), σ value, training sample size, etc. We also implemented the four different weighting techniques in order to compare the performance between them and to have a direct comparison between the K-NN and FRNN algorithms.

In general, the likelihood of a random season prediction for a test point would only yeild a correctness percentage of around 25%, finding a version of both the K-NN and the FRNN algorithms that were able to correctly predict the season in which a photograph was taken when given the location in which it was taken around 80% of the time was a great success.

References

[1] Zhou, Bolei, et al. “Recognizing City Identity via Attribute Analysis of Geo-tagged Images.” Computer Vision-ECCV 2014. Springer International Publishing, 2014. 519-534.

[2] Shamma, David A. “One Hundred Million Creative Commons Flickr Images for Research.” (June 24th 2014). Retrieved Nov 16, 2014 from `http://yahoolabs.tumblr.com/`

post/89783581601/one-hundred-million-creative-commons-flickr-images-for

[3] Daum III, Hal "A Course in Machine Learning". Retrieved in (Oct, 2014) from http://ciml.info/dl/v0_9/ciml-v0_9-all.pdf

[4] California Map from http://www.mapsofworld.com/usa/states/california/lat-long.html

[5] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, Ian H. Witten (2009); The WEKA Data Mining Software: An Update; SIGKDD Explorations, Volume 11, Issue 1.

[6] Bay, Stephen D. "Combining Nearest Neighbor Classifiers Through Multiple Feature Subsets." ICML. Vol. 98. 1998.

[7] "ICSI Works with Yahoo Labs and Lawrence Livermore Lab to Offer Analytics Tools for Over 100 Million Flickr Images and Videos". Retrieved Dec 09, 2014 from https://www.icsi.berkeley.edu/icsi/news/2014/07/icsi-works-with-yahoo-llnl

[8] Weyand, Tobias, Jan Hosang, and Bastian Leibe. "An evaluation of two automatic landmark building discovery algorithms for city reconstruction." Trends and Topics in Computer Vision. Springer Berlin Heidelberg, 2012. 310-323.

[9] Wang, Josiah K., et al. "A Poodle or a Dog? Evaluating Automatic Image Annotation Using Human Descriptions at Different Levels of Granularity." V & L Net 2014 (2014): 38.

[10] Hauff, Claudia. "A study on the accuracy of Flickr's geotag data." Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval. ACM, 2013.

[11] Gallagher, Andrew, et al. "Geo-location inference from image content and user tags." Computer Vision and Pattern Recognition Workshops, 2009. CVPR Workshops 2009. IEEE Computer Society Conference on. IEEE, 2009.

[12] Dickerson, Matthew T., Eppstein, David "Algorithms for proximity problems in higher dimensions." Computational Geometry: Theory and Applications, Volume 5, Issue 5, 1996. 277-291.