WiFi Access Point Mapping and Embedding

Rebecca Mercer

Abstract

This project aims to build a network of campus WiFi access points from time series data. The concept of multidimensional scaling is utilized to discover spatial relationships between access points from temporal data.

ECE 697 Final Project Report
Electrical and Computer Engineering
University of Wisconsin-Madison
August 2021

Contents

1 Introduction

Graphs can be used to represent many things in a typical persons daily life. A graph could represent how someone moves about their community, how the covid-19 virus spreads across the country, even the internet is a graph... In this project I explore methods for producing graphs representing the relative spatial relations between points using measures other than distance to determine how close or how far they are. Two interesting examples of building graphs from times series data of points are trying to build a network of sensors in a smart home and trying to build a network of WiFi access points around campus.

1.1 WiFi Access Point Problem

There are approximately 11,000 WiFi access points around the UW-Madison campus and the time series log of connections between personal devices and the access points. The goal is to build a graph of campus from the temporal data associated with the access points. The movement of devices around campus can be interpreted in such a way to draw conclusions about the spatial relationships of access points or buildings. Through producing a graph of campus we may be able to cluster buildings by their true proximity or by how they are related by how students and faculty move about campus. It would be interesting to see if it is possible to build an accurate map of campus from the data. It would also be interesting to see how closely buildings/areas of campus are related based on peoples movement. Determining how closely related buildings/areas are could be useful in contact tracing efforts to alert people to how covid-19 spreads about campus.

1.2 Home Automation Problem

In my family's smart home there are over 200 sensors (motion sensors, under floor pressure sensors, thermal sensors, etc). Understanding how the sensors are related to each other can help with the houses control of the lights and heating. One of the concepts of the house that would make use of this is: ideally one would be able to walk throughout the house with all the lights turning on ahead of them and off behind them without having to touch a light switch. Using the time series data collected when each sensor is triggered to build a map of sensors and further understanding how people move about the house and better predict how many people are in a given room will increase the efficiency of the homes lighting control. Previous work with the home data has done N-gram analysis to determine how long it typically takes to move from one sensor to another. There is a challenge in the fact that multiple people can and likely are moving around the house at the same time. This leads to impossible conclusions like it taking a few seconds to go from a room in the

basement to a room two floors away. Exploring other methods of determining the spatial relationships of sensors could be useful in this project.

2 Background

This project explores multidimensional scaling methods for representing how points are related in space from the time series data relating the points.

2.1 Multidimensional Scaling (MDS)

Multidimensional scaling finds the embedding or coordinates of the objects of interest from their pairwise distances. This makes MDS an unsupervised learning method. (7) MDS takes a square dissimilarity matrix made up of the pairwise distances and a number of dimensions *N*. *N* is the number of dimensions for the resulting coordinates to be placed in. MDS is often used in Machine learning for non-linear dimensionality reduction. (7) In these applications the distances represent the dissimilarity or proximity in a higher dimensional space. From this matrix embeddings in *N*-dimensional space are produced to best recreate the original data or represent the data in a lower dimensional space. (10) In other situations MDS can be used to produce an embedding that best represents the dissimilarities, which can be measured in many different ways.
There are many different categories of MDS including metric multidimensional scaling and non-metric multidimensional scaling.

2.1.1 Dissimilarity Matrix

The *n*x*n* symmetric dissimilarity matrix, *D*, represents the pairwise distances.

$$D = \begin{bmatrix} d_{1,1} & d_{1,2} & ... & d_{1,N} \\ d_{1,1} & d_{1,2} & ... & d_{1,N} \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ d_{N,1} & d_{N,2} & ... & d_{N,N} \end{bmatrix}$$

The pairwise distance $d_{i,j}$ can be the Euclidean distance between x_i and x_j. Otherwise it can be some measure of the dissimilarity or proximity between x_i and x_j. (8) Entries $d_{i,j}$ where $i == j$ must be 0. There should not be 0's where $i \neq j$ since this would mean x_i and x_j are the same point.

2.1.2 Classic MDS

The classical Multidimensional Scaling algorithm is the general form and takes dissimilarity matrix *D* and does not make assumptions about how the dissimilarity is measured. The algorithm first, converts the dissimilarity matrix to the gram matrix K. The matrix K is a Gram matrix computed by $K = -0.5H(D)^2H$ where H is the centering matrix ($H = I - \frac{1}{n}ee^T$) and e is a vector of ones. K is a positive semi definite matrix and can be expressed as X^TX
The aim is to minimize the difference between the distances given in D and the distances between the resulting coordinates.

$$min_Y \sum_{i=1}^{n}\sum_{j=1}^{n}(d_{ij}^{(X)} - d_{ij}^{(Y)})^2$$

where (X) is the distances from D and (Y) are the distances between the coordinates.

$$min_Y \sum_{i=1}^{n}\sum_{j=1}^{n}(x_i^Tx_j - y_i^Ty_j)^2$$

$$min_Y Tr(X^TX - Y^TY)^2$$

By Singular Value Decomposition: $X^TX = V\Lambda V^T$ and $Y^TY = Q\hat{\Lambda}Q^T$. Y^TY is also a positive semi definite matrix so there exists a closed form solution to this optimization problem. The resulting coordinates Y can be calculated as such $Y = \hat{\Lambda}^{1/2}Q^T$. (1)
The error of classical mds is called the strain ($strain = Tr(X^TX - Y^TY)^2$).

2.1.3 Metric MDS

The Metric Multidimensional Scaling algorithm is used when the elements of the dissimilarity matrix *D* are the true Euclidean distances. (8) The elements are calculated as follows where p is the number of dimensions in space of the original data.

$$d_{i,j} = \sqrt{\sum_{k=1}^{p}(x_{ik} - x_{jk})^2}$$

The embeddings of $(x_1, ..., x_n)$ in N-dimensional space are computed as in the classical algorithm. The common loss function for metric MDS is the *stress*. *Stress* is the residual sum of squares between the Euclidean distance and distances between the resulting coordinates.

$$Stress_D(x_1, x_2, ..., x_N) = \sum_{i \neq j = 1,...,N}(d_{ij} - ||x_i - x_j||^2)^{1/2}$$

2.1.4 Non-Metric MDS

The Non-Metric MDS (nMDS) algorithm is used when the dissimilarity matrix entries are not the true Euclidean distances, but some other measure. nMDS also accounts for situations where entries that are zero not on the diagonal are missing values. The algorithm handles this by relaxing the distance constraint and assuming the elements of *D* are computed by a function, f, of the true distance and some associated error. nMDS aims to find the relationship between dissimilarities and "true" or Euclidean distances. And use this to calculate the resulting coordinate embedding. (3) The loss function that is minimized in nMDS is *stress*.

$$Stress = \sqrt{\frac{\sum (f(d) - x)^2}{\sum x^2}}$$

Where d is the vectors that make dissimilarity matrix *D* and $f(d)$ is the monotonic transformation of d. x are the point distances between the calculated coordinates.
The key parts of the nMDS algorithm are as follows...
`1) initialize coordinates` *(The initial values may be random or may use the results of metric MDS)*
`2) calculate distances x between the coordinates`
`3) find optimal f(d)`
`4) update coordinates to configuration that minimizes stress`
`5) repeat 2-5 until some criteria is met` *(ex: stress less than some threshold, iterations exceeds max iterations, etc.)*
nMDS is more flexible to various measurements of dissimilarity and leads to nonlinear embeddings. This algorithm is not guaranteed to converge to an optimal solution due to multiple local minima.

2.1.5 Applications of MDS

One famous application of Multidimensional Scaling was for mapping how colors are perceived by humans.(4) In this experiment researchers had people rank how similar they thought two colors were. The averages of these rankings were used as the entries of the dissimilarity matrix and MDS was used to produce coordinates for each color. The plot visualizing the results produced the well-known color wheel. This is an example of using a measure besides distance and the result being used to visualize data capturing the relationships between data points.
In machine learning, MDS is often used for dimensionality reduction. The goal is to represent high dimensional data in a lower dimension, if a low dimension that captures relevant features of the data exists. Dimensionality reduction through MDS can be useful

for many instances with this goal including digital signal processing of images and audio and visualizing large networks. (1)

3 Campus WiFi Access Point Mapping

This project aims to visualize the spatial relationships between WiFi access points or groups of access points from time series data collected from the UW-Madison campus access points through multidimensional scaling.

3.1 Access Point Data

The University WiFi access point set was collected during spring semester 2021. Any time a mac address connects to any of the 11k WiFi access points around campus the following is recorded:

`start_time`: time connection is established
`installation_id`: identifier of the WiFi access point
`station_id`: identifier of the mac address

Each day of data consists of approximately 2 million data points.

3.1.1 Preprocessing

The first step when working with the data is to get it in a form that is useful for measuring the dissimilarity between access points. SQLite was used to join all pairs of data points with the same `station_id` and compute the difference between the `start_time`'s as `time_apart` between the two `installation_id`'s as `ap_1` and `ap_2`. Then points that capture the time between the same two `installation_id`'s are combined, saving the average time apart. This process was first completed for one day of data and later for a full week of data.
For the full week of data further considerations were taken in the preprocessing. On campus most buildings can be reached from one another within an hour of walking. To accommodate this only time differences under 2 hours apart were kept. 2 hours is somewhat of an arbitrary choice. Limiting the time apart considered is better than letting times that are unreasonable days apart influence the data, but setting the limit on the time apart is more challenging. Consider two access points down the hall from each other and two access points miles apart. A time apart of close to 2 hours would affect the accuracy of the average time apart on the two points very close together much more than the points miles

apart. In the full week of data the minimum time apart, `min_time`, was also recorded for each access point.

3.1.2 Dissimilarity Matrix

For this project multiple variations of dissimilarity matrices were experimented with to find the best way to measure the "proximity" or "dissimilarity" between the access points with the given data.
Starting with one day of data I experimented with building dissimilarity matrices including all access points, subsets of the most common access points, grouping access points by associated building, and scaling each measure by the frequency of each building to building connection.
Some connections between access points were very common and some connections were not at all common. In order to deal with certain connections flooding the data set I explored methods for scaling the dissimilarity measure. What I used was an adaptation of the natural language processing (NLP) method TF-IDF. I used $tf * idf$ calculated as follows to scale each entry in the dissimilarity matrix.

```
tf = term frequency (number of times the connection
          between building a, b appears)
   idf = inverse document frequency = log(N/df)
N = total number of connections between all buildings
  df = document frequency (sum of freq of union of
connections that building a and building b each have)
```

With one week of data the following dissimilarity matrices were built in the python script `DissimilarityMatrix_2021_04_01_week` (see more information in Appendix).

1. `'ds_mat_b_avg.csv'`
    - size: 207 x 207
    - measure of dissimilarity: average time in seconds apart (over connections between all access points of one building and all access points between the other building)
2. `'ds_mat_b_min.csv'`
    - size: 207 x 207
    - measure of dissimilarity: minimum time in seconds apart (between connections between all access points of one building and all access points between the other building)

3. `'ds_mat_b_avg_scaled.csv'`
    - size: 207 x 207
    - measure of dissimilarity: the values of matrix 1 each scaled by the tf-idf
4. `'ds_mat_b_min_scaled.csv'`
    - size: 207 x 207
    - measure of dissimilarity: the values of matrix 2 each scaled by the tf-idf

The same four variations of the dissimilarity matrix were created for access point to access point connections as well.

3.1.3 MDS Approach

The Scikit-learn MDS package was used to produce the embeddings of the buildings and access points. (9) Each dissimilarity matrix built was an input to a MDS fit_transform function. Since I manually created dissimilarity matrices using variations of measurements, I used the precomputed parameter. The starting points were randomized. The metric parametric of this function was set to false to indicate that this problem requires non-metric MDS algorithm. The non-metric MDS algorithm makes sense for this problem because of the missing data in the form of zeros in the matrix and because I cannot be sure that time apart is proportional to the Euclidean distance. Time apart seems to be a good measure of distance, but with the noise in the data and the fact that there exist different methods of transportation it cannot be assumed to be a perfect measure of distance, rather a useful measure of dissimilarity.
MDS returns the coordinates for the access points or the buildings of the given dissimilarity matrix. These coordinates were then visualized a few at a time. Some random sets of buildings or access points were visualized, but for this report we visualize buildings that someone familar with the UW-Madison campus may recognize.

3.2 Discussion of Results

3.2.1 Ground Truth Data

The ground truth location of the access points is given by the longitude and latitude. Using this ground truth data to verify the relative positioning of access points does not yield great results as the ground truth data is noisy, each point comes with an accuracy that is often very low, and the data does not take into account height as it only provides longitude and latitude.

3.2.2 Stress

Stress, the measure of how well the results of MDS fit the input are useful for tuning the parameters of the MDS algorithm. However, when it comes to validating the results of the embedding stress only tells us how well it fits the input. The results of the MDS algorithm are only as good as the dissimilarity matrix. Most of the improvements in the results come from careful data preprocessing and data collection.

3.2.3 Visualization

Many of the results can be visualized in the Appendix and more can be visualized in the code base (see Appendix).

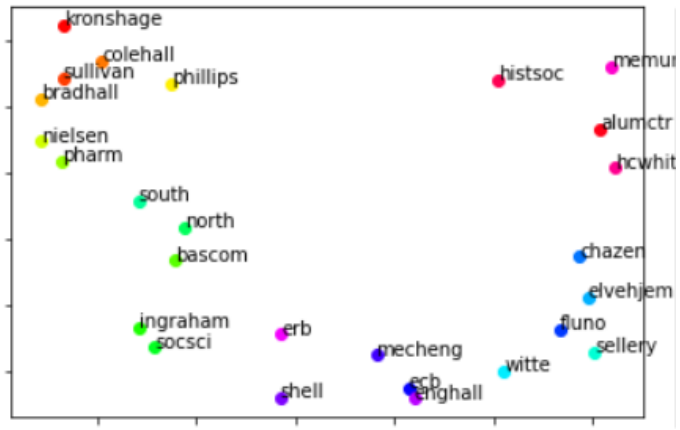

Figure 1: Visualization of Results of MDS using Scaled Minimum Time Dissimilarity Measure

Here we see an example of some of the buildings from the result nMDS using the scaled minimum time as the measure of dissimilarity. On the next page you will see the annotated results along side the campus map to visualize how these results compare to an actual 2D map of these campus buildings.

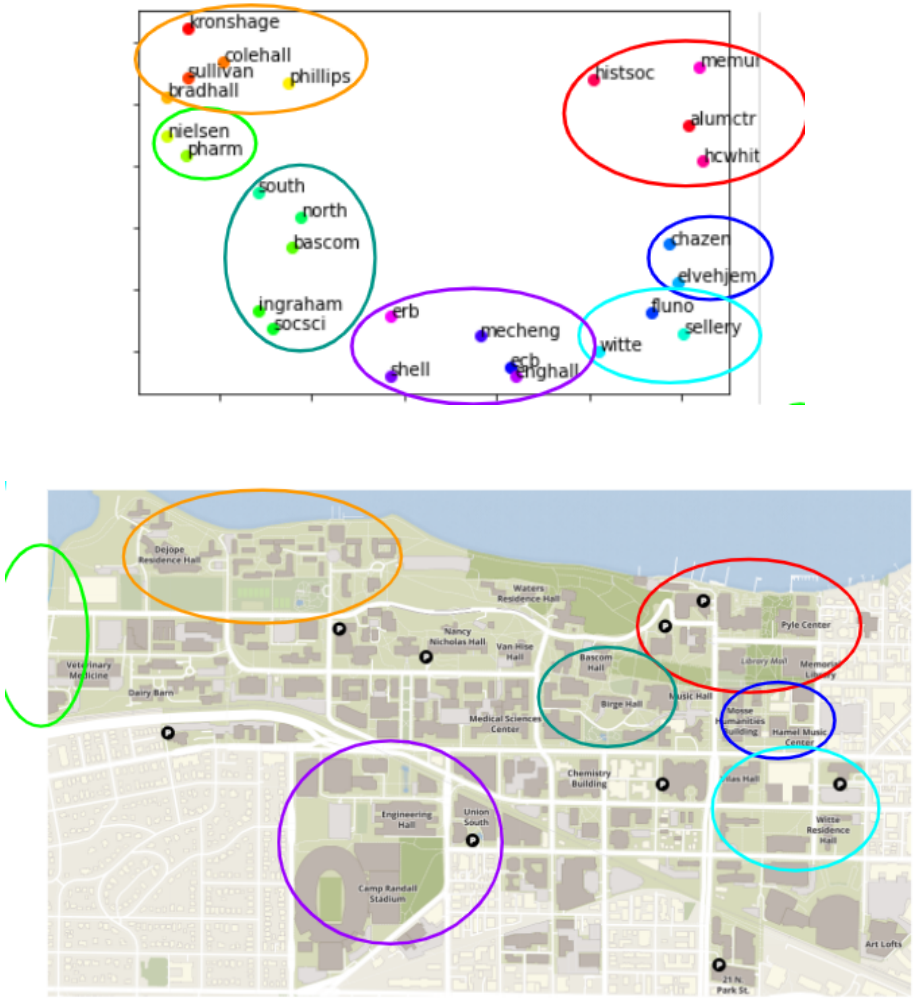

Figure 2: Visual Comparison of Results with Campus Map Annotated

Here we can see that many of the groups of buildings are relatively close to one another and give some resemblance to the actual layout of the buildings.

4 Conclusion

Multidimensional scaling has many applications including calculating relative coordinates of points given some measure of pairwise distance between the point and representing data in a lower dimensional space. There are some ways to improve the results of MDS within the algorithm, however the results are still only as good as the data used to produce them. In dimensionality reduction, the usefulness of the results of MDS are dependent on their existing an N-dimensional structure that represents the higher dimensional data. When using some measure of dissimilarity the results are dependent on there existing a relationship that holds between the values of the measure and the Euclidean distances.

When it comes to processing the campus data it was interesting to see that grouping the access points by buildings successfully made up for the ”holes” in the data; as there were many missing connections between access points. Adjusting the dissimilarity measures by their frequency using my adaptation of the NLP method TF-IDF helped prevent very common connections from overriding the connections of uncommon data points. In choosing the dissimilarity measure minimum time seemed to better represent the Euclidean distance than the average time. By using the minimum time we throw away a lot of data. In order to get a better measure, while still preserving more of the data it would be interesting to look into using the log time or seeing what can be learned from the individual distributions of time.

This project allowed me to explore a machine learning method with many different applications that I may have not otherwise been exposed to and highlighted the importance of data collection and data preprocessing. In any statistical or machine learning model, the most important step is the preprocessing of the data. Careful data preprocessing lends itself to how accurate the results are to real life and the fairness of a model. As machine learning is being used in many areas that impact society data preprocessing is an important step that can have a lasting effect.

5 References

[1] Ali Ghodsi. (2006) *Dimensionality Reduction A Short Tutorial*.

[2] Agarwal et al. *Generalized Non-metric Multidimensional Scaling*.

[3] Buttigieg PL, Ramette A. (2014). *A Guide to Statistical Analysis in Microbial Ecology: a community-focused, living review of multivariate data analyses*. https://mb3is.megx.net/gustame/dissimilarity-based-methods/nmds

[4] Ekman, G. (1954). *Dimensions of color vision*.

[5] Joseph Imperial. (2019). *The Multidimensional Scaling (MDS) algorithm for dimensionality reduction*. https://medium.datadriveninvestor.com/the-multidimensional-scaling-mds-algorithm-for-dimensionality-reducti

[6] Michael W. Trosset. (1993). *The Formulation and Solution of Multidimensional Scaling Problems*.

[7] Nadja Herger. (2017). *Visualising the model space with unsupervised learning*. https://nherger.github.io/posts/mds-hierclust

[8] NCSS Statistical Software. *Multidimensional Scaling*. Chpt: 435.

[9] Pedregosa et al. (2011). *Scikit-learn: Machine Learning in Python*. JMLR 12. pp. 2825-2830.

[10] Stephen P. Borgatti. (1997). *Multidimensional Scaling*. http://www.analytictech.com/borgatti/mds.htm

[11] Stephanie Glen. *Multidimensional Scaling: Definition, Overview, Examples*. From StatisticsHowTo.com: Elementary Statistics for the rest of us! https://www.statisticshowto.com/multidimensional-scaling/

[12] Wikipedia. *Multidimensional Scaling*. https://en.wikipedia.org/wiki/Multidimensional_scaling

6 Appendix

6.1 Code Base

`https://github.com/rmercer2/ece697`
Here you will find the code base associated with this project and instructions if you wish to run part of this project yourself.
The key files that are used to go from the raw data to the visualizations of the embeddings produced by MDS on the various dissimilarity matrices are as follows:

1. `'DataPreprocessing_2021_04_01_week.ipynb'`
    - This notebook contains the work using SQLite to produce the `'2021_04_01_week_associations_times.csv'` file.
      This file has columns: ap_1, ap_2, total_time, frequency, and min_time.
2. `'DissimilarityMatrix_2021_04_01_week.ipynb'`
    - This notebook produces the various dissimilarity matrices from the `'2021_04_01_week_associations_times.csv'` data.
    - The matrices are all outputted as .csv files named according to which matrix they represent.
3. `'MDS_2021_04_01_week.ipynb'`
    - This notebook contains the functions for normalizing the data, running MDS to return coordinates, and visualizing a given set of points.

6.2 Example of Figures Produced

Figure 3: One day of Data: Access Point to Access Point Measured with Average Times

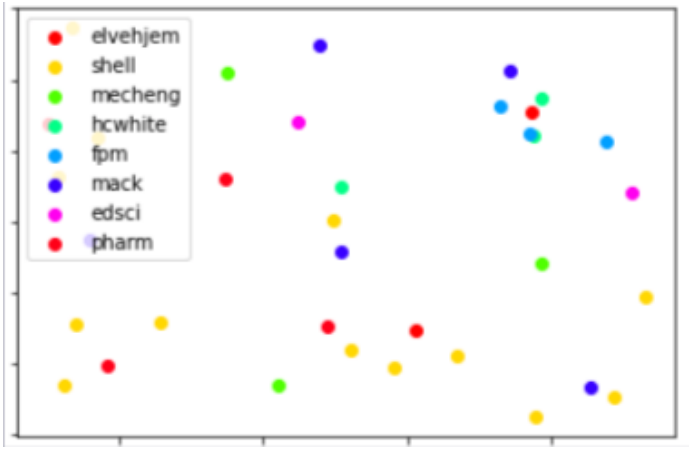

Figure 4: One day of Data: Building to Building Measured with Average Times

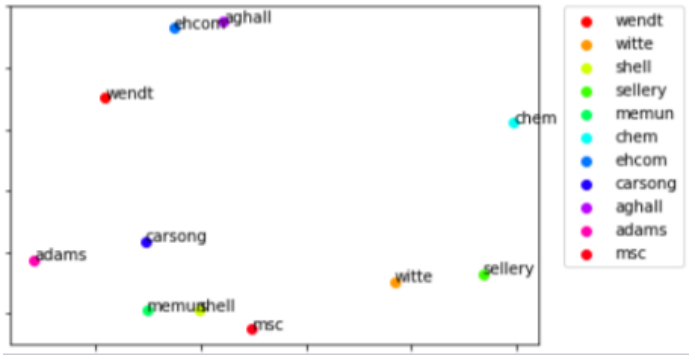

Figure 5: One day of Data: Building to Building Measured with Average Times and Scaled by Frequency

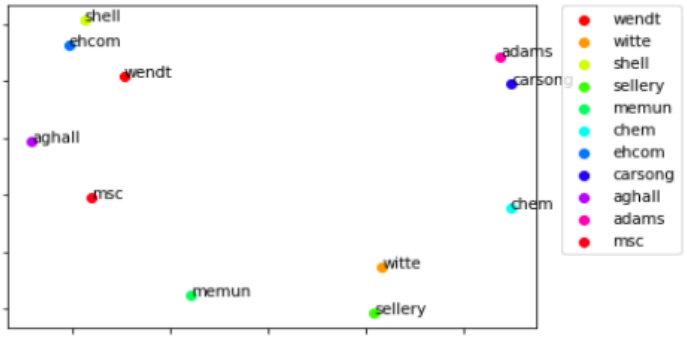

Figure 6: One week of Data: Building to Building Measured with Average Times

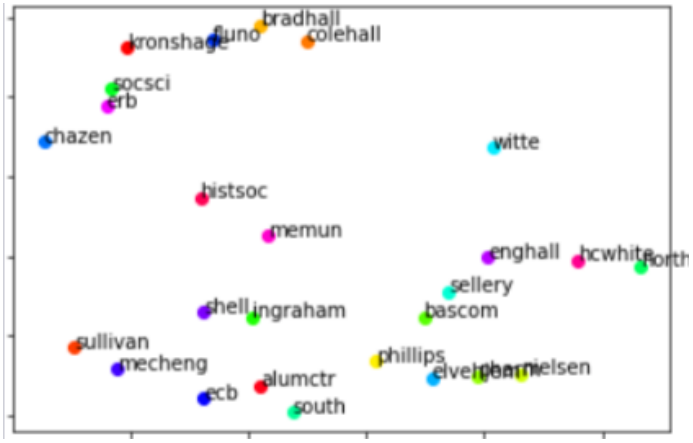

Figure 7: One week of Data: Building to Building Measured with Average Times and Scaled by Frequency

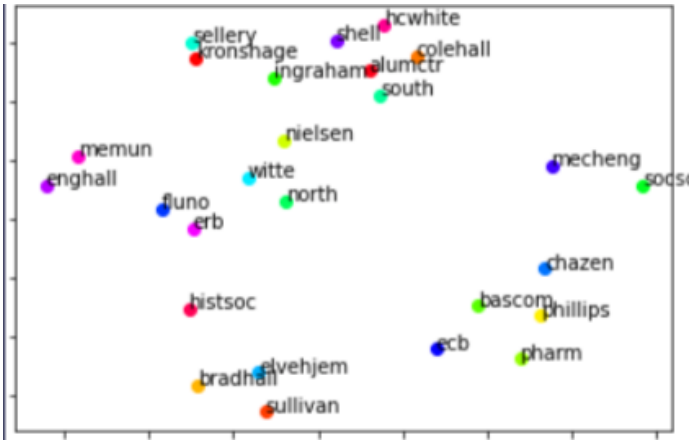

Figure 8: One week of Data: Building to Building Measured with Minimum Times

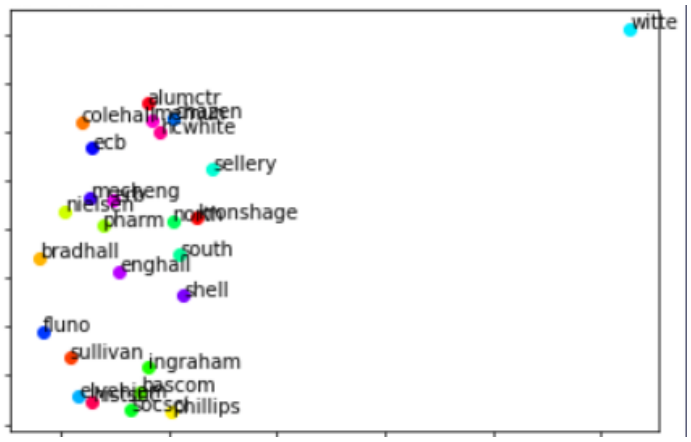

Figure 9: One week of Data: Building to Building Measured with Minimum Times and Scaled by Frequency