# Assignment 4 Report: Data Clustering

Annelise Wittenberg & Cody Stammer

([wittenberg.18@osu.edu](mailto:wittenberg.18@osu.edu); [stammer.3@osu.edu](mailto:stammer.3@osu.edu))

## Approach

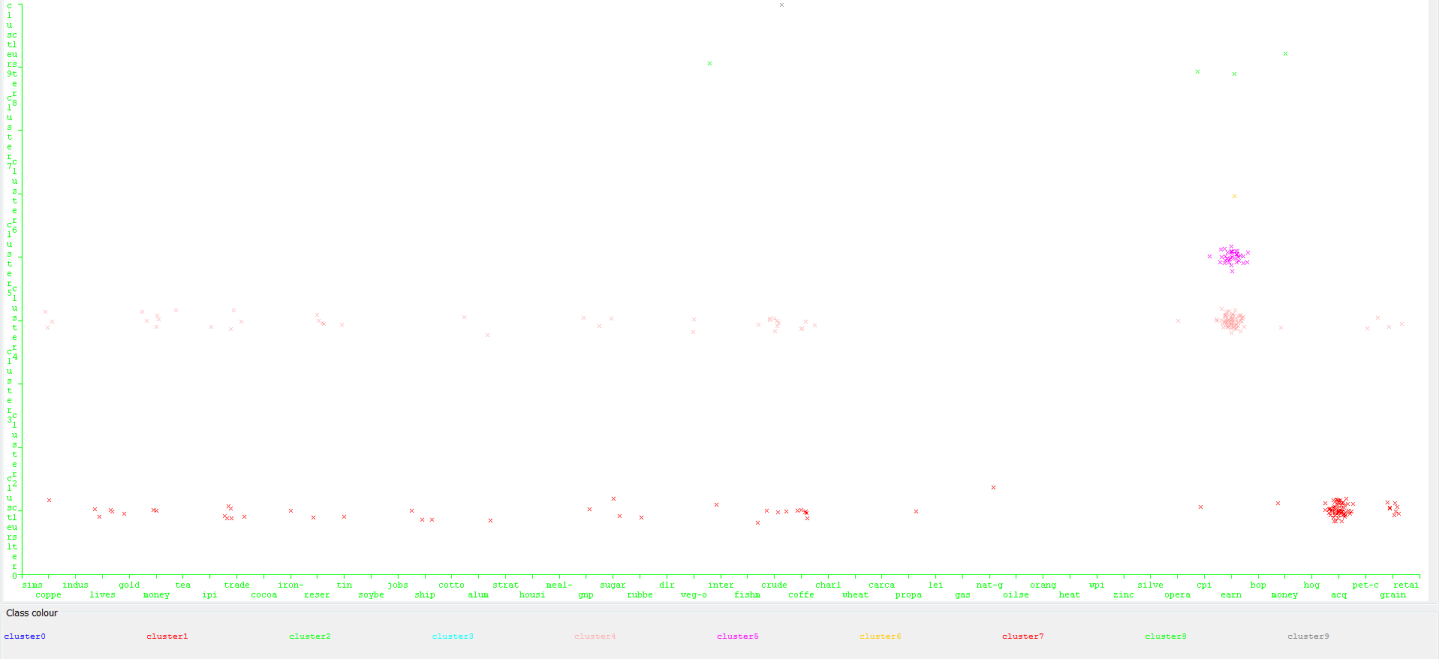
For this assignment, we first eliminated any topic and its associated articles where the topic occurred 3 or less times. We chose to do this filtering because having those values made it nearly impossible to find a good cut point for the dendrogram in hierarchical clustering. This step removed just over 2000 articles from our data set.

We chose to do hierarchical and K Means clustering, and Euclidean and Manhattan distance metrics. We used Weka once again to evaluate the clustering algorithms and distance metrics. We used Orange to evaluate the hierarchical clustering after realizing that Weka consistently clustered all data into one cluster, even when the parameters were changed.

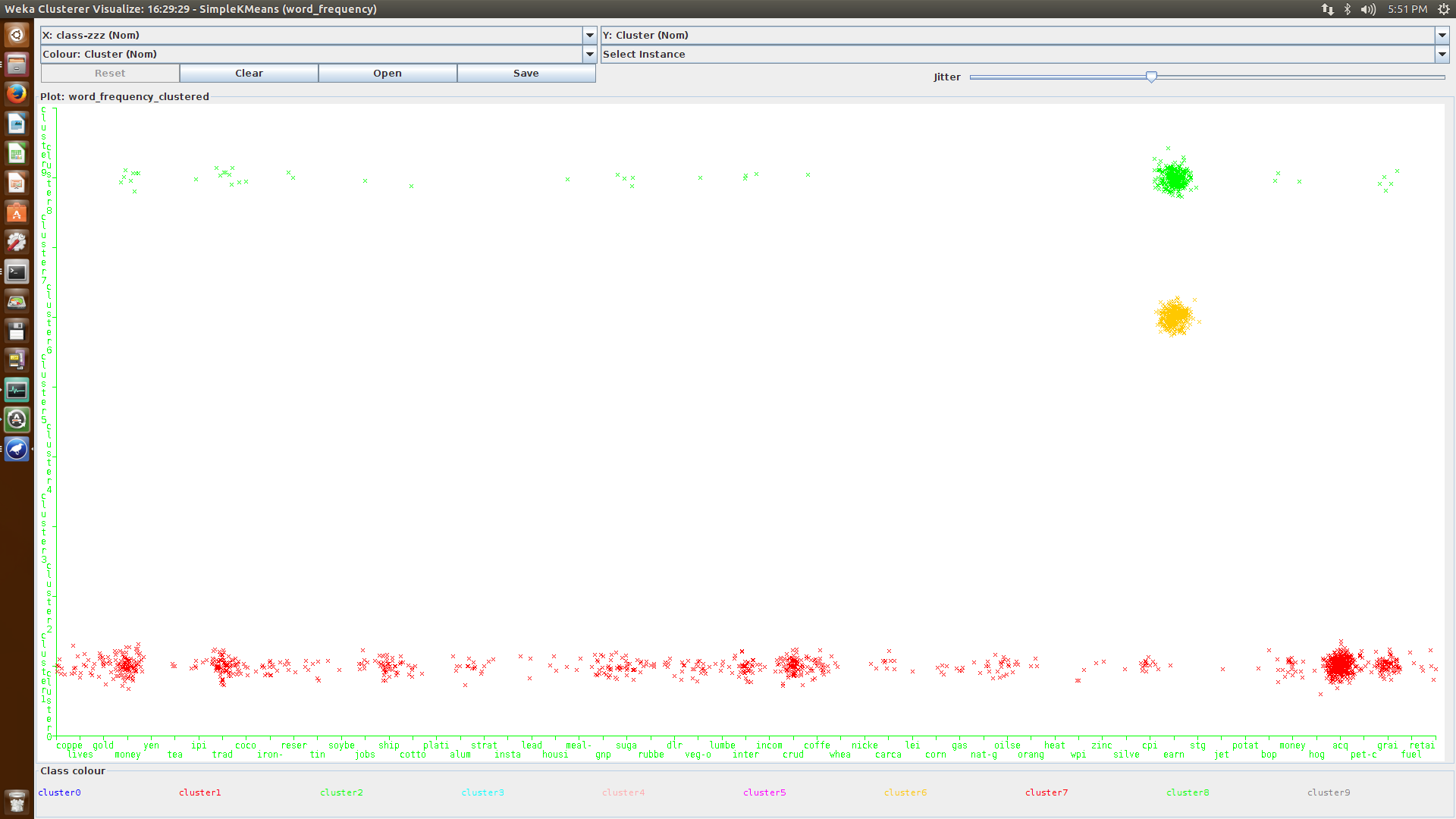
## Results

The table below contains data for our runs on the entire dataset.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Hierarchical**  **Euclidean** | **Hierarchical**  **Manhattan** | **K Means**  **Euclidean** | **K Means**  **Manhattan** |
| **Time** | 3180.2 s | 2214.08 s | 2970 s | 1722.58 s |
| **# Clusters** | 8+ | 10 | 10 | 10 |

In the K Means (Euclidean) approach with 10 clusters, 40% of the data points when to cluster 1, 44% to cluster 4, 13% to cluster 5, and 2% to cluster 8. The remaining 1% was spread out over the remaining clusters, with a very small amount of data in each cluster. With 80/20 training to testing split, it was no surprise that it took about 49.5 minutes to complete. This time is consistent with classification time in the previous lab. The algorithm completed in 14 iterations. The picture below shows a visual distribution of where the clusters appeared on the graph. There is a very sparse green cluster and gray cluster near the top. The resulting arrangement is relatively skewed towards clusters 1 and 4.

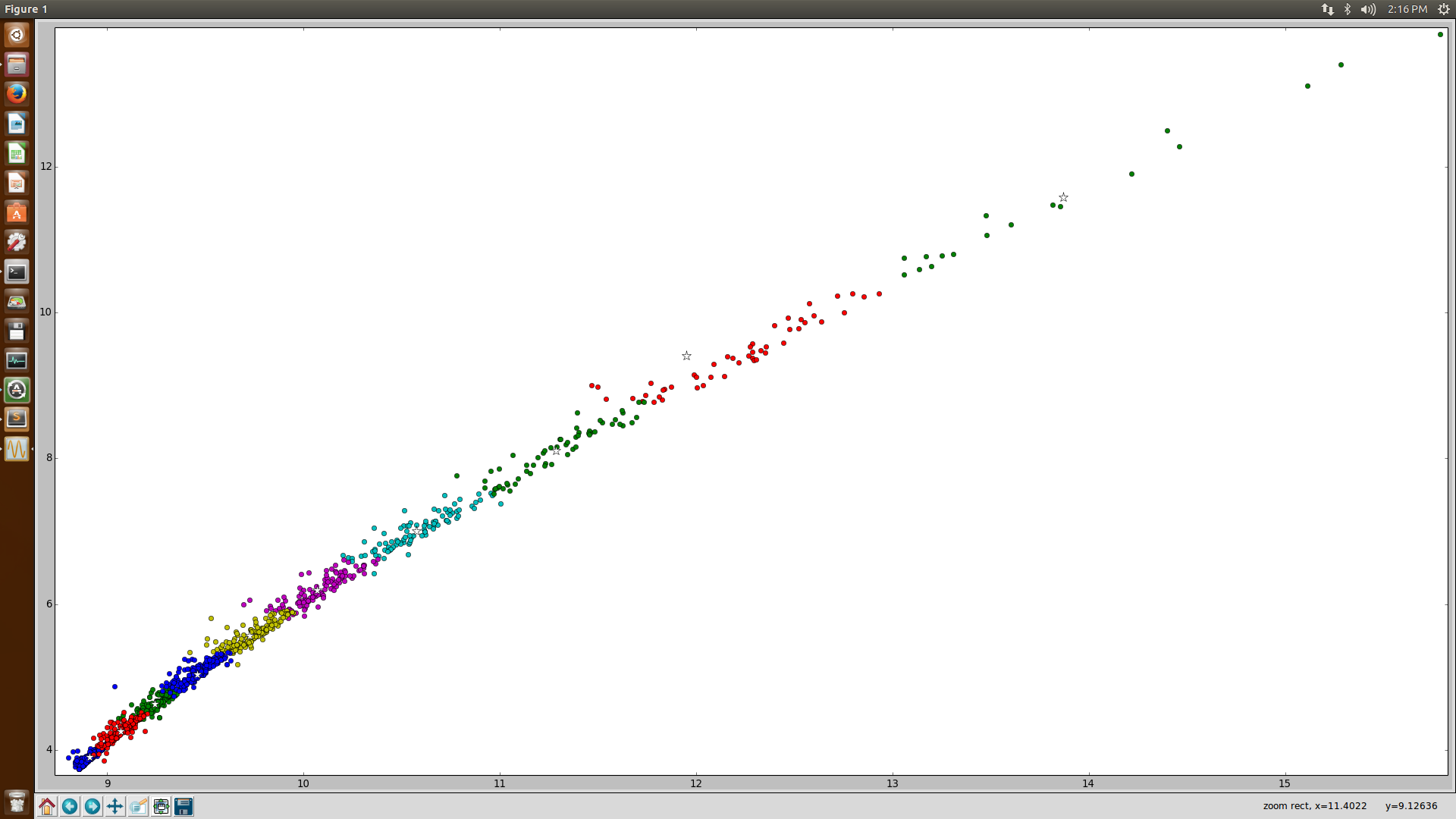
In the K Means (Manhattan) approach, it took less time to complete the clustering, nearly half of the time, and it only took 8 iterations to complete. The distribution of the clusters is a lot more dramatic. 60% of the data went to cluster 1, 21% went to cluster 6, and 19% went to cluster 8.



The hierarchical (Euclidean) approach took 3180.2s to run. RAM usage during this process exceeded the 11 Gigabytes allocated to the machine and flowed into swap space. A cut was made on the resulting dendrogram was cut resulting in 8 clusters containing more children trees. This result is not extremely desirable as many articles are not contained in a cluster other than themselves. The entropy of this cut would be near zero as many articles are contained in their own clusters. A view of this cut can be viewed in the “dendrogrammanhattan.SVG” when opened in a browser that can render SVGs. The clusters containing children trees are highlighted in various colors. The highlighted clusters seem to be relatively accurate but do not encompass a majority of the data which is why the low entropy as not ideal.

The hierarchical (Manhattan) approach took 2214.08s to run. The amount of RAM usage increased at a very slow rate and stayed roughly around 13GB throughout the latter half of the process. When this algorithm was run in Weka, it clustered every instance to cluster 0. This skew is very awful since k-means showed how better results could be obtained. Since everything was put into one cluster, the entropy was 0. This may sound like a good value for entropy, but it is known that not all data points should belong to the same cluster. When we tried to load the data to Orange and ran the same kind of analysis, the entropy was slightly higher but the distribution of data points in the clusters was much better. 10 turned out to be a good number of clusters for this data set as well because by looking at the diagram below, it is easy to see that the clusters contain mostly words of the same topic label. See “dendrogrammanhattan.svg” to view the clustering.

The clustering.py could also produce a kmeans clustering plot on a smaller set of data shown in the image below. A larger set would result in error. The Orange implementation of the kmeans seemed to have a better division of clusters and more desirable entropy of data points. However, it does not seem that Orange is able to process the large amount of data contained by our data preprocessing. Various methods to reduce the data set were done but were still not able to be processed through python scripting. This is why we then resulted to using Weka to cluster the data using kmeans.



## Issues

We first ran into issues when deciding what methods to use for the clustering algorithms. Weka has a good interface for showing timing of each run, and the distributions of the clusters, but does not show plots and dendrogram well. Orange has a very easy GUI to use with a good interface for hierarchical clustering, but it does not provide any kind of visual representation for k-means clustering results. The GUI also does not time the processes. When we implemented the lab using the Orange library for python, we ran into issues with runtime. The program ultimately timed out and wouldn’t complete. Our implementation thru python would have the desired results but the large amount of data cause the program to never finish execution and stall.

## Assumptions

For this assignment, we assumed that testing with an 80/20 split would be sufficient for each type of clustering, since this type of split had the best results in the classification lab. Since we would be gathering data for two runs of each clustering algorithm (one for each distance metric), we assumed that doing only one type of split would be sufficient. We wanted to avoid having to run four more tests (one for each distance metric for each clustering algorithm) since the overhead of running one test was so large, and crunch time was approaching.

We also assumed that Weka’s and Orange’s implementation of hierarchical and k-means clustering would be relatively correct. They provided fairly similar results to our python implementation.

## Work Distribution

Clustering Algorithms - Annelise

K Means Plot - Cody

Dendrogram - Annelise

Report – Annelise and Cody

Testing – Cody

README - Annelise