Tyler Percy

Homework 3 Analysis

Analysis Objective

For this homework, the objective was to compare the performance of K-Means using different distance metrics as the way of calculating the SSE. Although K-Means usually used with Euclidean distance, I found a package that allowed you to specify your preferred distance so that I could use Jaccard and Cosine as well. The problem with using these other metrics is that K-Means is not guaranteed to converge for all metrics. Cosine will converge and also uses the mean of the cluster as the centroid but I did have to raise the minimum distance between centroids required for the K-Means to stop from 1e-6 to 1 for Jaccard because it was running infinitely (and even still it takes around 30 seconds).

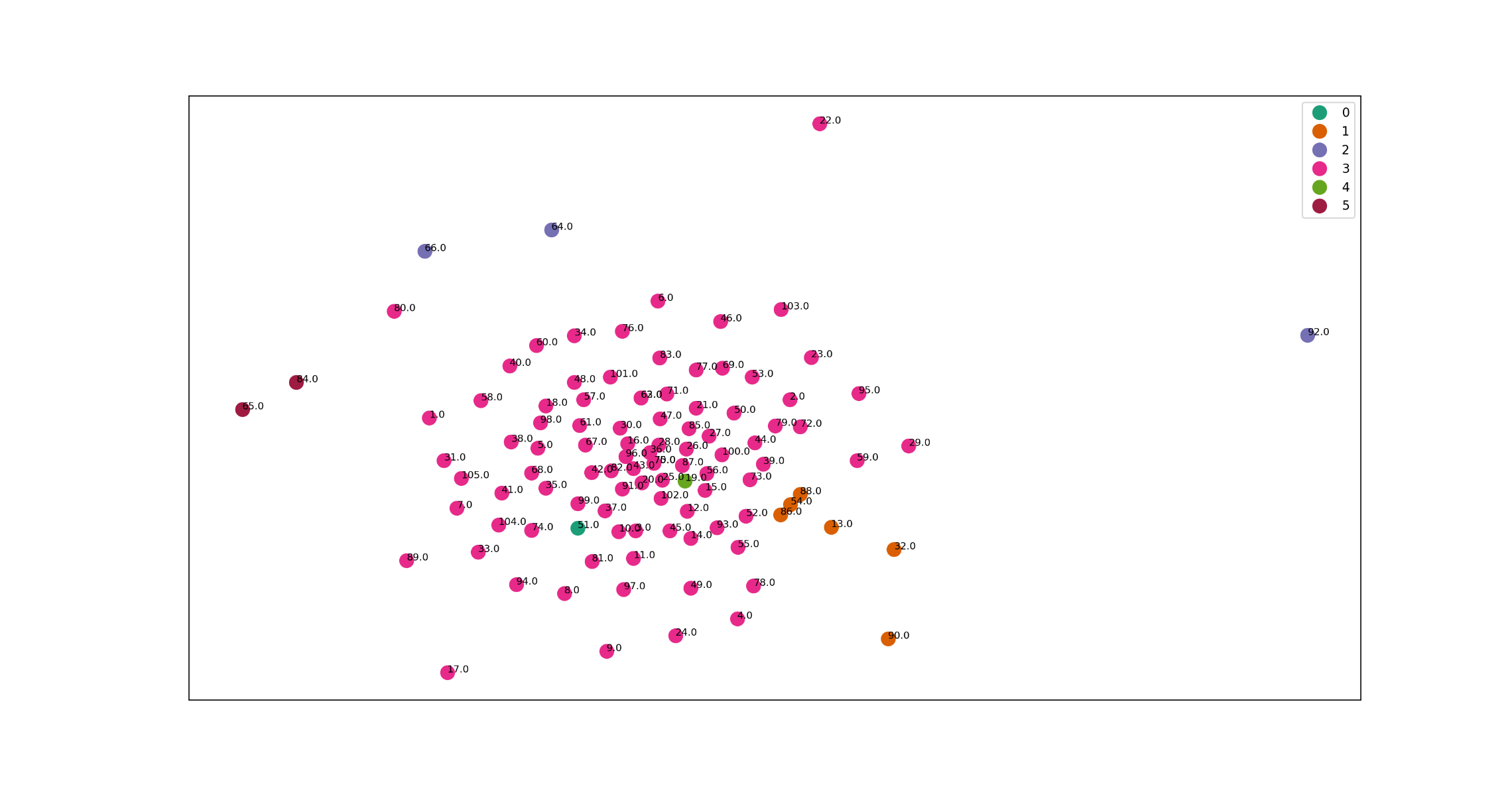
Process Description

I used Python and the nltk package for K-Means and the distance metrics. I used sklearn for transforming the article data matrix to a lower dimension that could be graphed in order to visualize the clustered articles. For making the actual graphs I used mapplotlib. Although the nltk did classify the data using K-Means, I had to create my own implementation to calculate the SSE for each cluster by looking at the results for the labels, the original data matrix for the values and then calculate their distance from the means. For the SSE results, I ran the program 10 times and averaged the results to make sure the output would be more consistent.

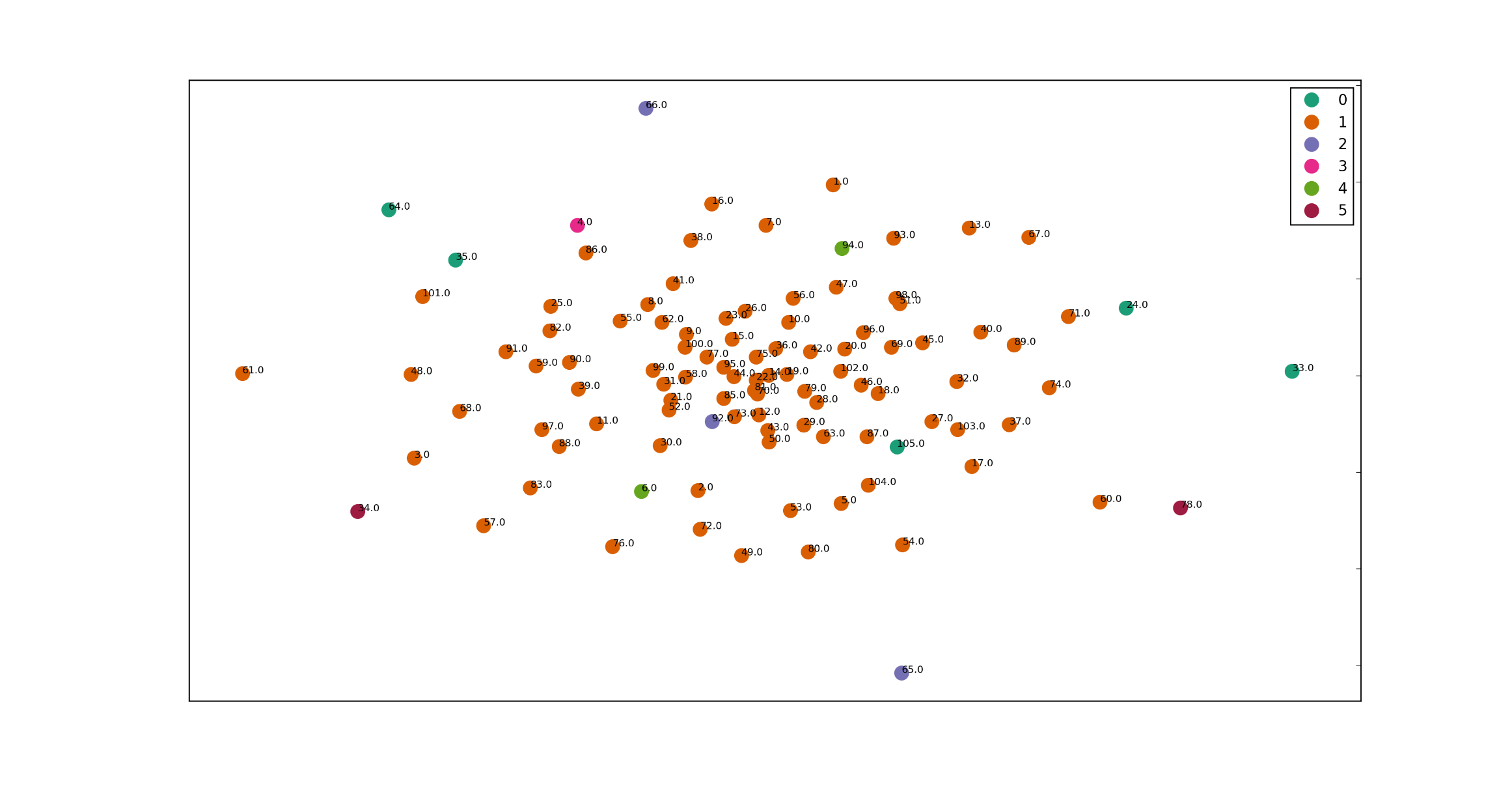
Overall K-Means Results

For Euclidean distance, the SSE was very large averaging around 110,000 while cosine was around 40 and Jaccard was about 100. I knew there would likely be some issues with the output for my K-Means due to the nature of the data set. I took the top 105 articles on CNN.com at the time and due to the recent election, this caused a large amount of the articles to be politically related. 45 of the 105 articles were put in the “politics” category and another 18 were from the “world news” category and 13 were from the “US” category. Because of the strong overlap in used in these categories, I did expect there to be some problems in properly identifying the differences between the clusters. The graphs reflect these problems as it shows that the majority of the articles are centered around a single large cluster with some points scattered away. The classifier would have worked much better with a lower k value but I followed the instructions of the homework and used k=6 for the number of categories. Overall, I was still happy with the results as they reflected my expectations knowing the data set going in.

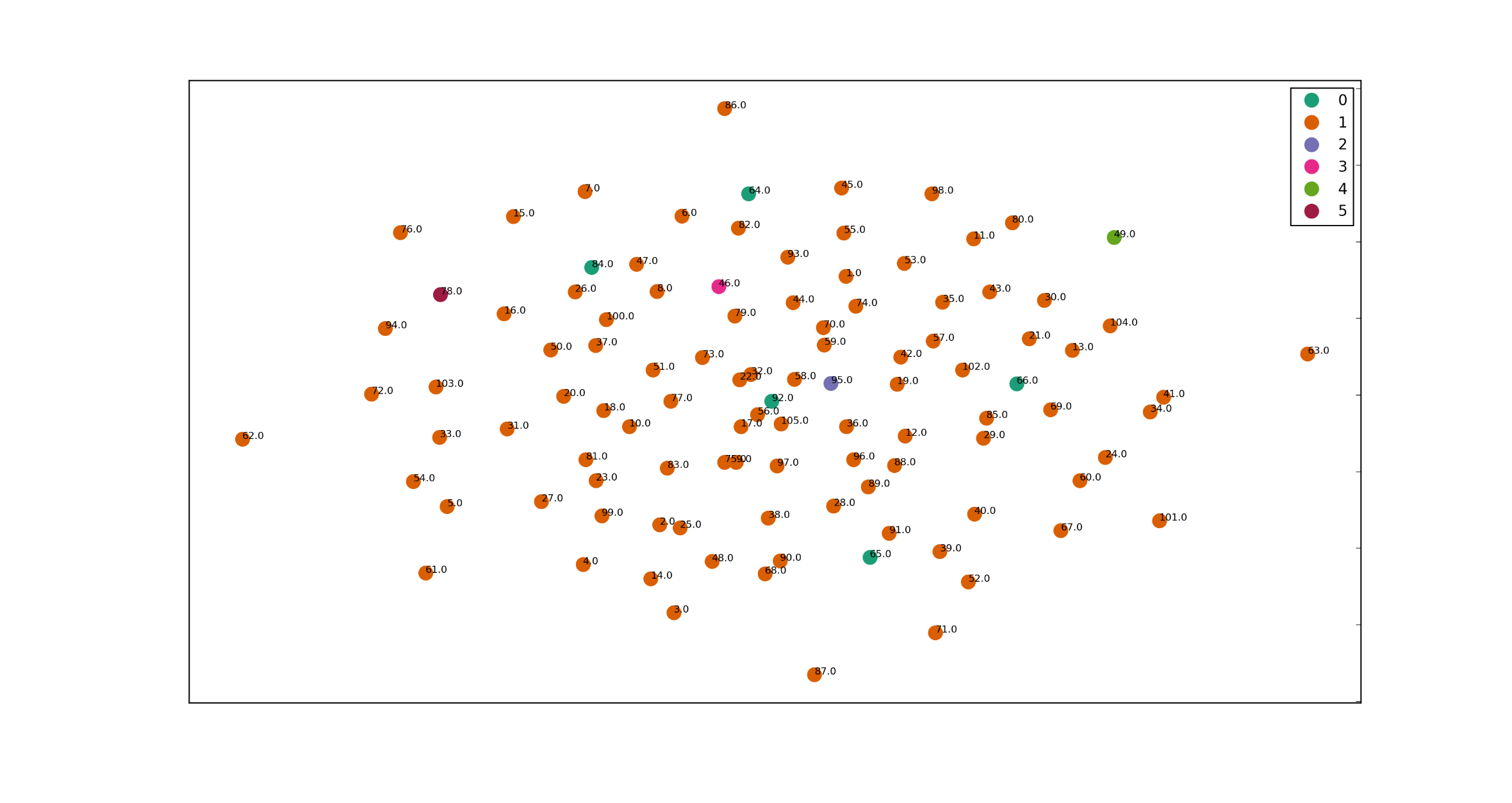
Graphs

This was my first time using any coding packages for visualization and I think that the graphs turned out very well and do a good job of showing the clusters. Although the article data had to be transformed to be 2 dimensional in order to graph the articles in this way. This explains some of the odd things you see such as cluster 4 being randomly its own point in the middle of cluster 3.

Euclidean Results

Although the cosine had a significantly smaller total SSE for the clusters, this graph does not show those particluar results very well so it may be lacking in that reguard. I believe that the cosine graph is more spread out because the cosine similarity is better suited for analyzing document frequency data compared to euclidean. This is why near the center the data points are very dense. These are the political articles that are very similar to each other. However, as you get away from the center, the articles are a bit more spread out from each other than the euclidean distance. This is because the cosine similarity does a better job picking up on the minor differences between the articles.

Cosine Results

Like cosine, the jaccard graph does not really show the results given by the SSE, as it is probably the worst looking graph. The points are pretty evenly distributed and there really aren’t any clear clusters except for one huge one in the middle (at least when represented in two dimensions). Jaccard can detect some similarity between articles shown by the tightly packed point in the middle of the graph but the rest of the articles are spread out evenly for the most part.

Conclusion

If I were to run this whole experiment again starting from homework one, I would use the TF-IDF matrix as well as normalizing the matrix to improve my results. I think each step of the process would have strongly benefited from that. Another major thing I would change is how I chose my articles. Having almost ¾ of all the articles being so similar lowered the performance of the classifiers, and with more clear differences, the accuracy would certainly improve. Despite these problems, I still learned a lot about data analysis theory in general as well as using python to perform the tasks.