Data Mining Programming Assignment 2 Report

Gregory Star -- starx013 -- 11/26/2018

Preprocessing

To do the preprocessing, I wrote a Python 3 script called ppScript.py which utilized HTMLParser and the Python 3 implementation of Porter's stemming algorithm provided on the linked website. Steps 1 – 9 of the text processing stage were performed using built-in python commands (except for stemming).

To my knowledge, I have followed all the provided preprocessing steps in the exact order they were presented, except I also added the removal of the word "reuter" (and removed articles that became empty after this step) since all bodies ended with this word. Zeren reported that his preprocessing script produced 8090 articles with 5618 unique tokens. My script produces 7472 articles with 5666 unique tokens. After discussing this in office hours, a hypothesis emerged for this discrepancy: namely, Zeren's script counts the top 20 most frequent articles before removing empty bodies, and mine removes empty bodies first. Unfortunately, the preprocessing script is so long and complicated that I can't be certain this is the cause, but it's quite likely. Since my preprocessed data seems to produce good results, and since visual inspection of my input files (and the clabel file) doesn't reveal any obvious mistakes, I've chosen to stick with my approach. The way I see it, it's probably better to remove too many articles rather than too few in terms of getting meaningful clustering solutions.

Methodology

For each combination of input file, clustering criterion, and number of clusters, 20 trials were performed. The results of the trial with the best criterion function were then taken and analyzed by computing their purity and entropy. Since there are three different input files, three different cluster numbers, and two different criterion functions, the program was run 18 times.

All tests were performed on a Dell Inspiron 15, 7000 series laptop (7th gen Intel i5, Nvidea GTX 1050). I tried to have only kcluster.java running for most of the tests, but since this laptop is my primary computer, this was not always possible to achieve. As a result, runtimes may be slightly variable; overall though, they seem to follow a linear trend which I'll discuss in the runtime analysis segment of the report.

Note that clustering run-times tend to be significantly quicker on a CSE Lab machine (>25% time reduction). If my scripts are run for grading purposes, faster times should be achieved.

Implementation Details

Data was read in from the input file and stored in several arrays. Since the vectors in the data set are sparse, it would not make much sense to store them as arrays of size 5600+. Instead, each vector was stored as an array of active dimensions and a corresponding array of component values. So for example, the vector [1, 0, 0, 0, 0, 0, 5, 0, 1] would be stored as [0, 6, 8] and [1, 5, 1] (in the data set, these vectors are all normalized). Each such vector was part of a double array, so that the entire data set could be accessed quickly. It should be noted double precision was used, though float precision would probably have sufficed and reduced run-time (largely due to page-faulting, but also due to double operation time vs float operation time).

For both the SSE and I2 criterion function, the first step was to initialize the centroids. Centroids were randomly generated based on the current seed, and are all normalized so that they start at the outside of the unit hyper-sphere.

The optimization for the SSE criterion function (standard K-Means) was implemented using a standard batch-update approach. First the entire data set is clustered, and then all of the centroids are updated.

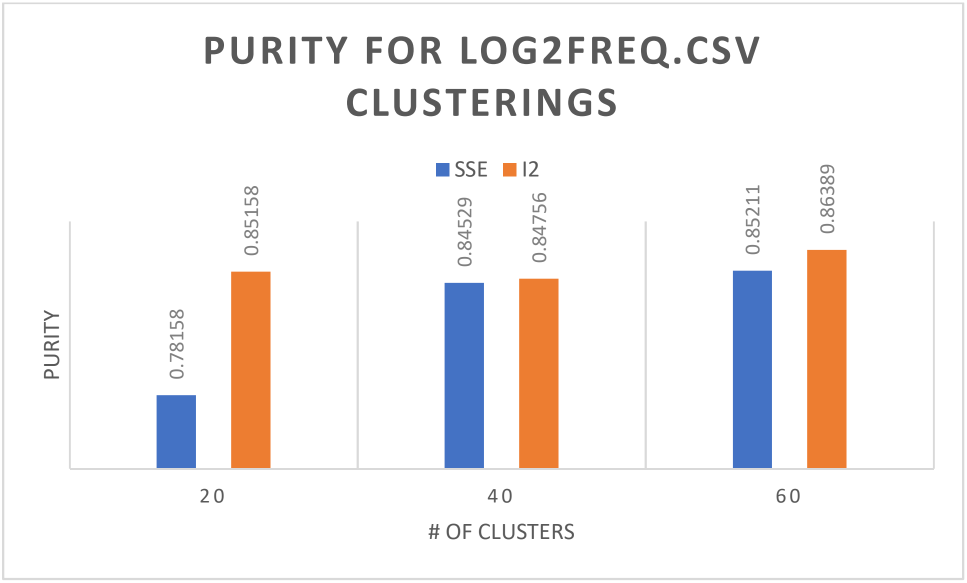
For the I2 criterion function, the alternate form of the function was optimized (sum of the class representatives). This was done in an online approach, where each vector is reclassified if that classification improves the total sum of the class representatives.

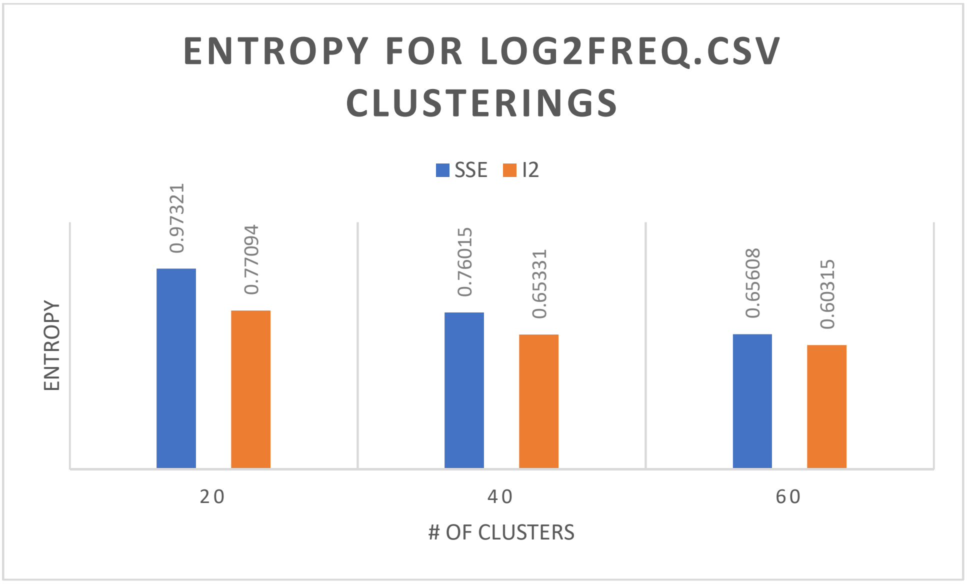
Note that for both standard K-Means and Spherical K-Means, I did not look for a convergence condition but rather just performed 10 iterations no matter what. At one point I was printing out the objective function at each iteration, and was finding very little convergence beyond 10. It was suggested in class that we cut off the program at 10, and given the runtimes I'm seeing for 20 trials, I think going any further would be excessive.

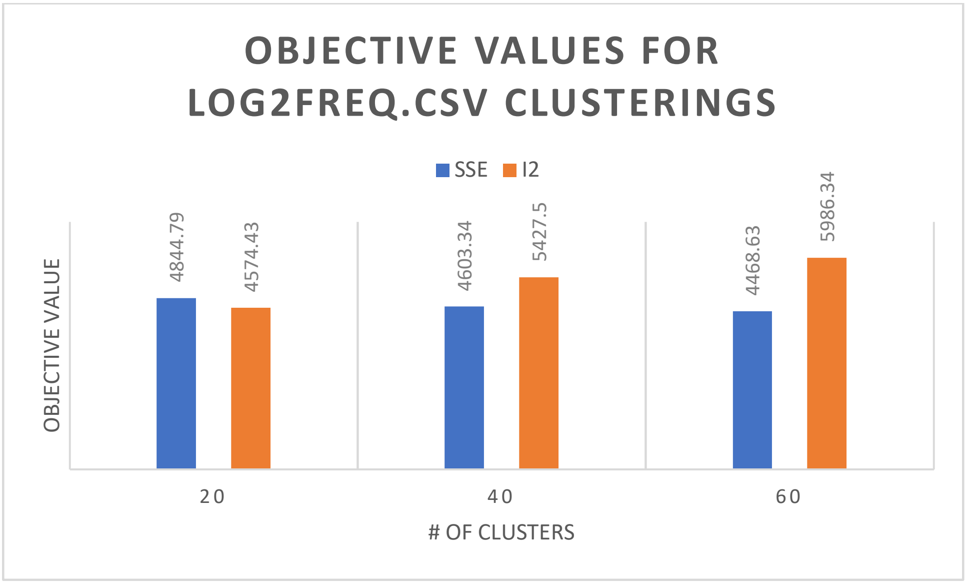
While purity and entropy calculations were relatively straight forward, I'd like to make note of the fact that I was having issues with entropy calculations when p_ij = 0. However, I was told to not add to the total entropy when this occurs, and taking that advice my results now appear sensible.

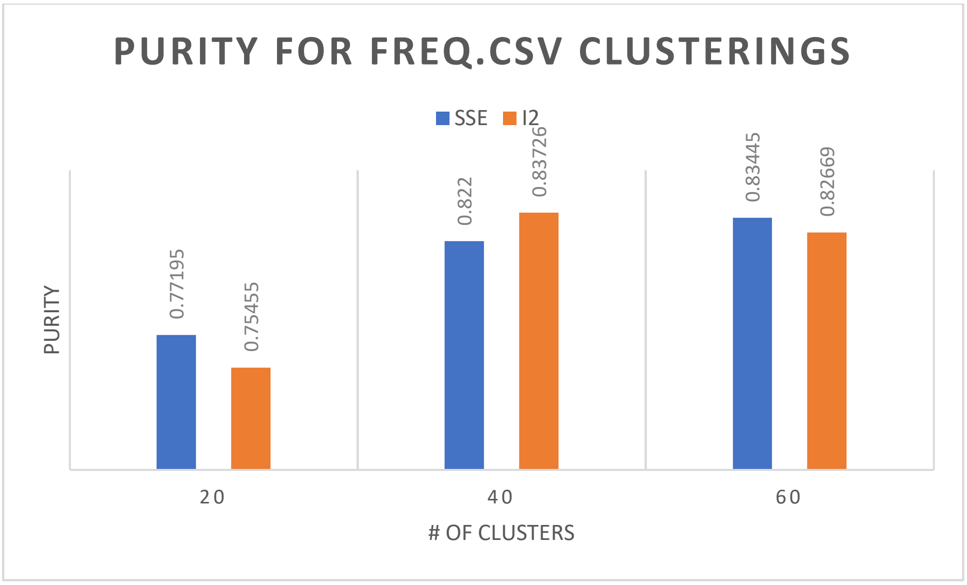
Clustering Results

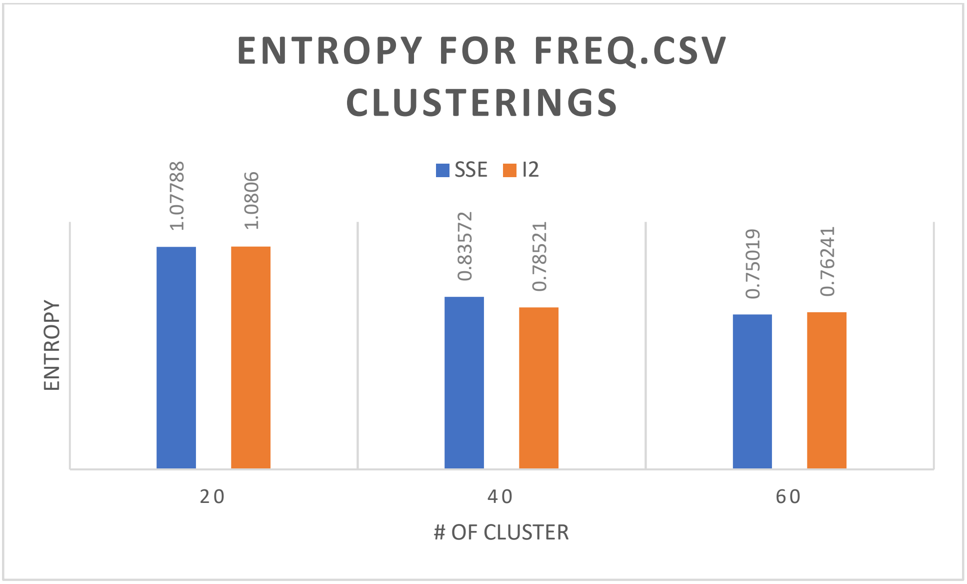
Below are bar charts for each of the three files (log2freq.csv, freq.csv, and sqrtfreq.csv); for each file, there is a purity graph, an entropy graph, and an objective value graph (*note that the objective values for SSE and I2 are put in the same graph, but high SSE values are bad and high I2 values are good*). This ends up being 9 graphs on the clustering solution quality.

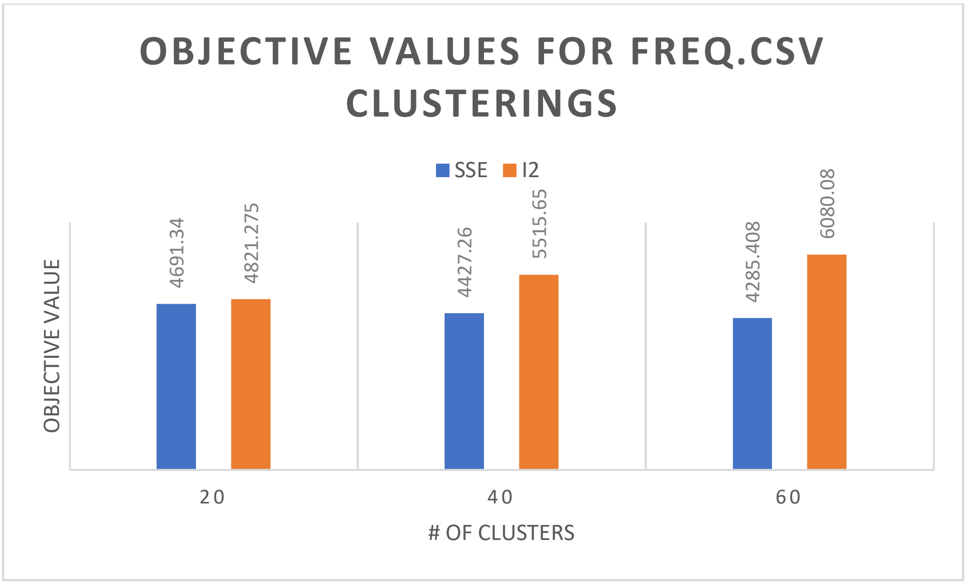


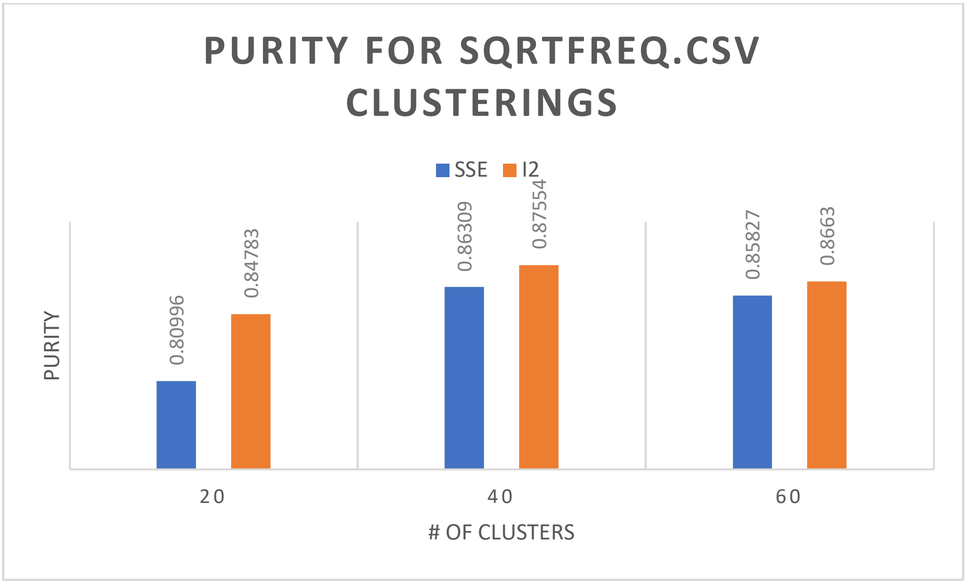

ENTROPY FOR LOG2FREQ.CSV CLUSTERINGS
SSE
I2
ENTROPY
0.97321
0.77094
0.76015
0.65331
0.65608
0.60315
20
40
60
# OF CLUSTERS

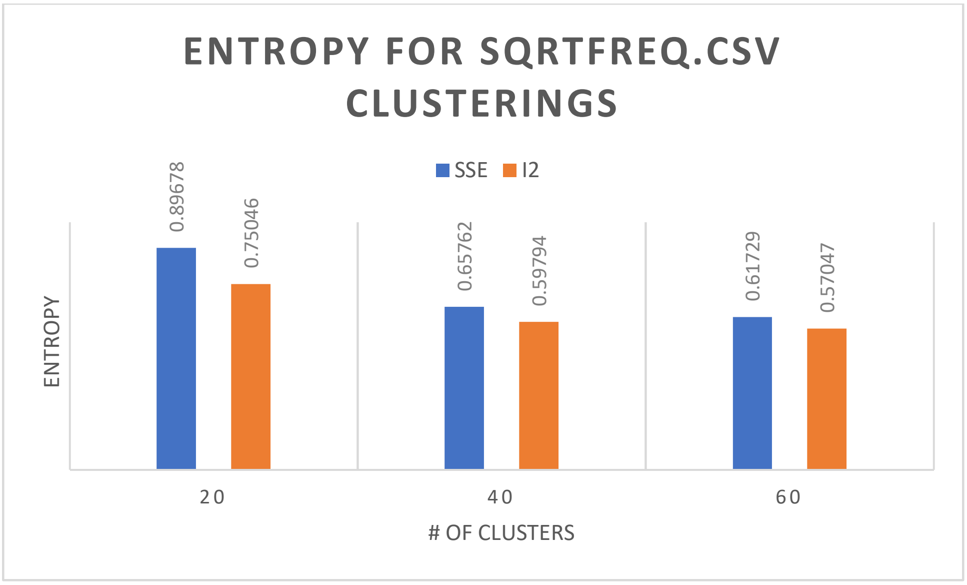

OBJECTIVE VALUES FOR LOG2FREQ.CSV CLUSTERINGS
SSE
I2
OBJECTIVE VALUE
4844.79
4574.43
4603.34
5427.5
4468.63
5986.34
20
40
60
# OF CLUSTERS

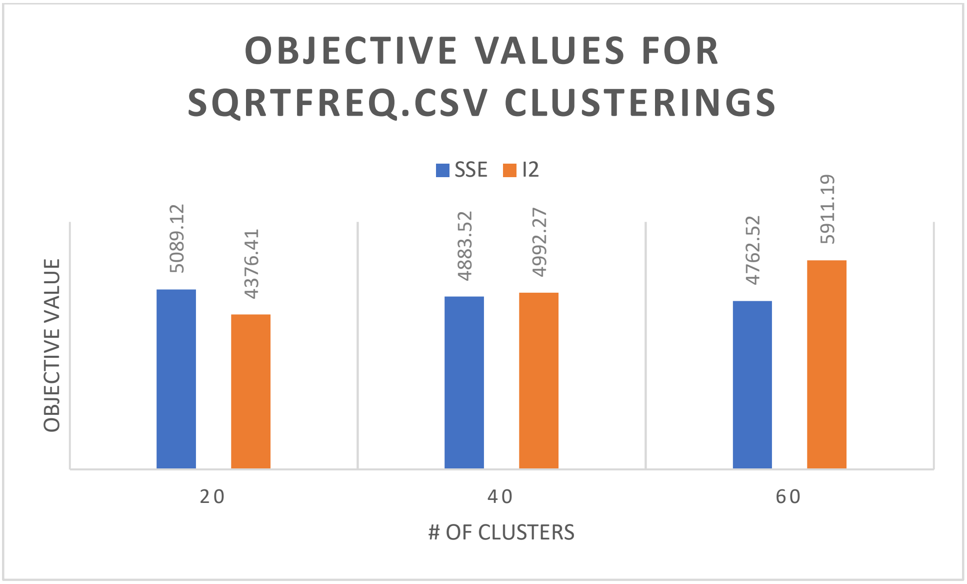

PURITY FOR FREQ.CSV CLUSTERINGS
SSE
I2
PURITY
0.77195
0.75455
0.822
0.83726
0.83445
0.82669
20
40
60
# OF CLUSTERS


ENTROPY FOR FREQ.CSV CLUSTERINGS
SSE
I2
ENTROPY
1.07788
1.0806
0.83572
0.78521
0.75019
0.76241
20
40
60
# OF CLUSTER


OBJECTIVE VALUES FOR FREQ.CSV CLUSTERINGS
SSE
I2
OBJECTIVE VALUE
4691.34
4821.275
4427.26
5515.65
4285.408
6080.08
20
40
60
# OF CLUSTERS


PURITY FOR SQRTFREQ.CSV CLUSTERINGS
SSE
I2
PURITY
0.80996
0.84783
0.86309
0.87554
0.85827
0.8663
20
40
60
# OF CLUSTERS





As expected, larger numbers of clusters produce better objective values, but this tells us little about the true quality of our clustering solutions. We can see from the data that while increasing the number of clusters tends to improve the purity, it is not guaranteed. However, the entropy decreases with larger cluster numbers across the board, which is promising.

My program prints out the best clustering solution for a given run, allowing me to manually inspect what's happening. Taking a closer look at the clustering solution reveals that some clusters appear to represent the same article label (topic). For example, it is common to see something like:

Cluster 1: 0, 0, 25, 0, 0, 0 1

Cluster 2: 0, 0, 10, 0, 1, 0, 0

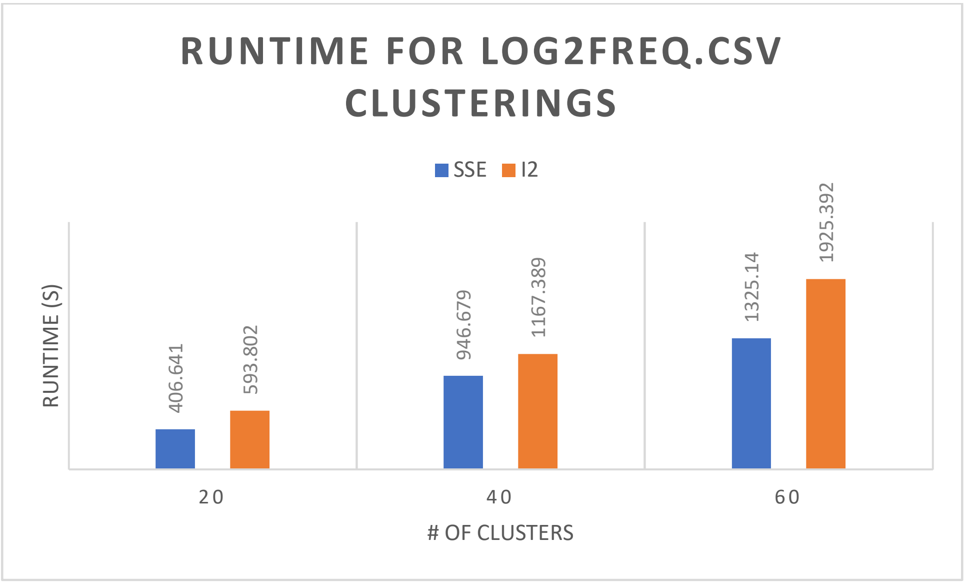
Since the purities tend to be high, we know that articles with the same label tend to be “similar”. However, the clustering solutions show that the algorithm is not necessarily clustering by label, but rather by term similarity. In order to see exactly what’s happening, the centroids of the solution would need to be printed, and manually inspected to see the top terms; this would’ve been an interesting addition to the assignment, but since it’s not in the instructions, I’ve chosen to leave this out.

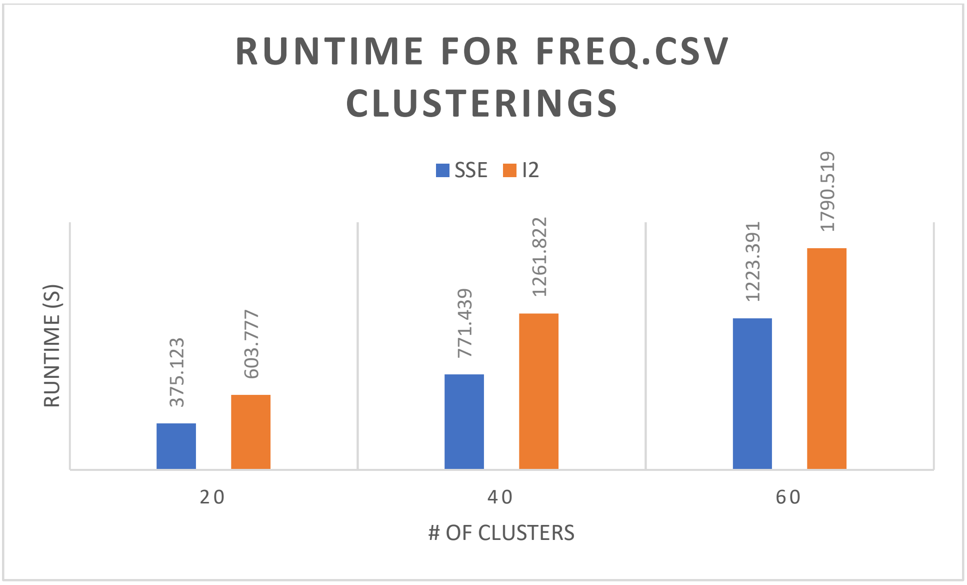
It should be noted that while SSE and I2 produce similar purities, I2 tends to produce better entropy values. If I had to choose a clustering algorithm for this data set, I’d go with I2. We can see that in some cases, the objective value for I2 goes above 6000, which is fantastic given that the highest objective value we could possibly get is 7472. This tells me that the I2 clustering solution is the closest to finding a “true” clustering of the data set.

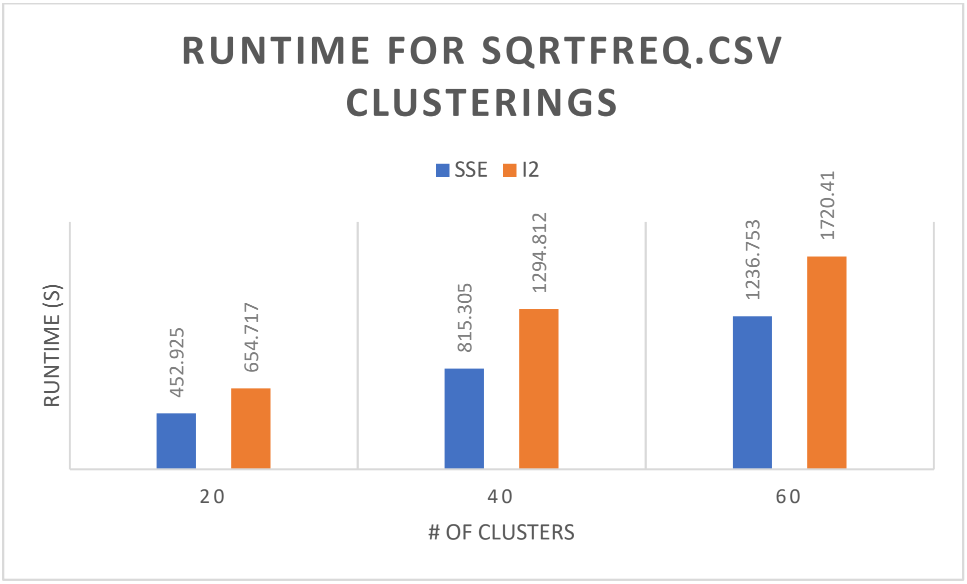
It is not clear from the data whether log2freq.csv or sqrtfreq.csv perform better, but what is clear is that freq.csv performs the worst; this is likely due to the extreme sparseness of the document vectors.

Clustering Runtimes

For both K-Means and Spherical K-Means we know that the runtime should grow as O(n * I * d * k), where n is the number of data points, I is the number of loop iterations, d is the number of dimensions, and k is the number of clusters; in this particular program, the runtime is actually O(n * I * d * k * T), where T is the number of trials. Since n, d, I, and T are fixed, we expect to see an approximately linear relationship between the time taken and the number of clusters. Plotting the results, we see that this is indeed the case.







As previously mentioned, runtimes would be faster on a CSE Lab machine, or under better testing conditions. However, we can see from the data that the I2 clustering solution is slower than the SSE solution. I am not sure if this is inherently true, or just poor implementation by me. Though I am not certain, I feel that I could've saved some time when deciding whether to swap a vector to a new cluster (though I did not see how) in the I2 implementation.

While the data seem to show some file representations perform better than others, I'm not sure we can conclude that this is really the case. When doing my trials, I ran all log2freq.csv tests, then all freq.csv tests, and finally all sqrtfreq.csv tests. For some of these tests I was away from my computer, but for others, I was browsing the internet. These figures should be taken with a grain of salt. However, that's not to say that it's impossible for different file representation to produce different run-times, just that we should be skeptical of this conclusion.