CSE 5243 Report for Lab 4

Arathi Mani & Roee Ebenstein

Description of Program

This program takes a vector file (the output file we created on homework 2) that contains a document number plus the number of times each of the titular words appears inside the document for each of the ~10k documents that were valid (i.e. contains a non-empty body tag and a non-empty title tag) and clusters the document using two algorithms and two distance metrics per algorithm. We also ran the K-mean clustering algorithm on the set of all documents, around ~22k, and calculated the entropies, variances, and errors.

We focused on implementing K-means clustering and hierarchical clustering with Euclidean and Manhattan distance formulas. The results were plotted and aggregated.

For the K-means clustering algorithm we ran the clustering algorithm twice - once only for the classified set, and once for all the valid documents - the results can be seen below.
For the hierarchical clustering it took too long to cluster all the documents, and after a while we ran out of memory, so we provide data only for the classified set.

In addition - all the code was written by us, we did not use any libraries.

Contents of Program

1. Lab3Loader.java - this file takes the output vector from Lab 1 and transforms it into a HashMap that maps the document number to an ArrayList of the vector values (i.e. the

"coordinates" of the point). From here, the formatted data is fed into the two clustering algorithms: K-mean and hierarchical.

2. KMeansClustering.java - this file actually performs the K-means clustering algorithm. Its private variables are the data (formatted in Lab3Loader), the number of centroids, and the convergence threshold under which the algorithm may cease. The last value tells you how small the distance a centroid update can be before you stop updating and computing new centroid values.
3. HierarchicalClustering.java - this file performs the Hierarchical clustering algorithm.
4. ResultsComputer.java - this file was used in KMeansClustering.java in order to compute the entropy and standard deviation based on the output files generated by the K-means clustering algorithm and the Topics file that was generated from the previous lab.
5. output-all.zip - this file contains all the valid data sets, for learning the cluster.
6. output-class.zip - this file contains only the classified valid data sets.

Team Split

What Arathi did: Hierarchy, debugged K-means
What Roee did: K-means, debugged hierarchy
Documentation, this document, and conclusion making have been done together.

Discussion

While running the K-means clustering algorithm, we noticed that when we have more clusters, the memory consumption is much higher and it takes more iterations for the cluster algorithm to stabilize. This results in a much longer running time for a higher k.

The side effect of it is limited scale. We can run in parallel the mean calculations, but the memory consumption is still high for more than 64 classes so the memory limit of the machine will limit the scalability of the algorithm. Scaling out to more machines is possible, but it will slow the algorithm tremendously since all the machines will have to "talk" a lot.

We also noticed that for K-means, it takes longer to run the algorithm using the Euclidean distance formula compared to the Manhattan distance formula. Although it takes longer to run, Euclidean doesn't necessarily guarantee a better result over Manhattan. For example, the entropy for 32 clusters using Manhattan distance is actually less than the entropy for 32 clusters using Euclidean distance (3.947 compared to 3.973).

Another noticeable behavior - the "all" data set increases the entropy tremendously, therefore we should only cluster based on the classified data. There are a lot of "dirty" files that creates less concise clusters.

Hierarchical clustering is a very slow process. Not only it is slow, it also consumes a lot of memory (about 300MB for 1000 papers, in our data set). Though it takes long, and a lot of memory - the huge benefit is that after the model is built - one can change between the levels without recalculating anything (something that is required in all the other methods we learned in

class).

There is no fast and easy way to parallelize the algorithm, for each iteration the input is the output of the previous iteration - therefore only the process within each iteration can be scaled. This could be possibly done by computing the distances in the matrix in sections and allowing a different computer core to work on each section before joining the threads together. We did not see any significant change performance wise among different distance measurement functions.

The results we got are pretty promising, a variance of 2000 for 16 clusters, and an entropy of 3, without any optimization is pretty good.

The results were very similar for K-means clustering as well. Around 16 clusters, we got a variance of about 2000 and entropies that were all less than 4. The results for hierarchical clustering was a little better than K-means overall in terms of looking at the topic entropy. The entropy for the k-means is about twice of the hierarchical clustering entropy, while the variance is about the same - it means that the output of hierarchical is closer to the real given one (we need to hold less information for being able to classify), though it is much slower.

For all cases, increasing the number of clusters decreased the standard deviation (or variance).

There are more trends noticeable in the tables and graphs below.

Results

The following combinations of distance algorithms and clustering algorithms were run.

K-Means Clustering and Manhattan Distance

Results

	4 Clusters	8 Clusters	16 Clusters	32 Clusters	64 Clusters
Running Time	13.077	17.841	30.646	74.588	182.433
Error Rate	331081.56	330655.75	329650.3	328264.78	322966.3
Standard Deviation	4032.5	3075.22	2238.8	1604.78	1125.81
Topic Entropy	3.961	3.965	3.958	3.947	3.976

Table 1: Results of K-means clustering using Manhattan distance on documents that had valid classes.

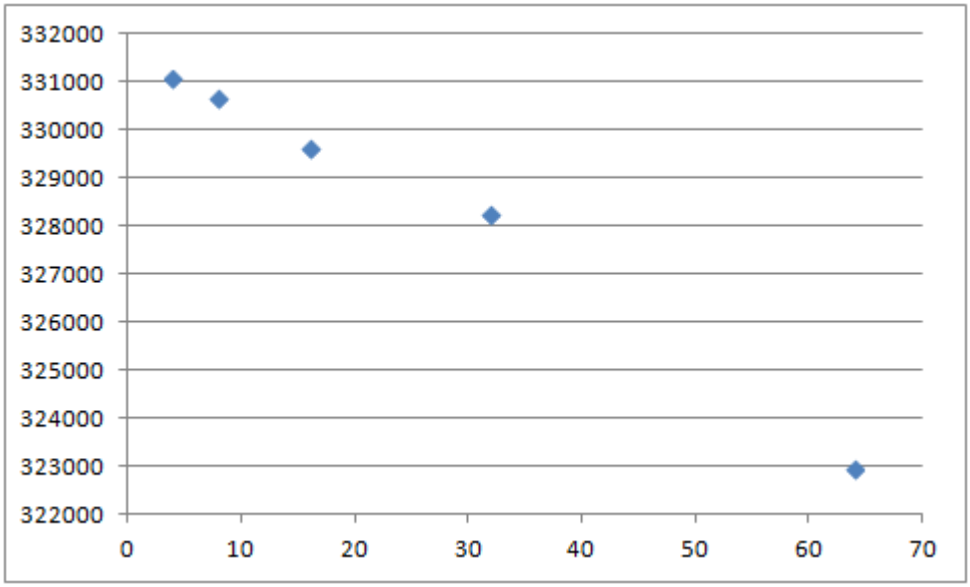

Figure 1: Error rate plot of K-means clustering using Manhattan distance on documents that had valid classes.

	4 Clusters	8 Clusters	16 Clusters	32 Clusters	64 Clusters
Running Time	11.289	32.492	68.999	205.242	337.101
Error Rate	644981.2	643996.44	642149.44	639368.1	633713.0
Standard Deviation	7820.355342789738	5971.8880271966755	4349.767493337287	3112.258416672439	2199.52690841681
Topic Entropy	7.742277662265584	7.735551459877068	7.741533240090853	7.720057981015784	7.721237587280536

Table 2: Results of K-means clustering using Manhattan distance on all valid documents.

K-Means Clustering and Euclidean Distance

Results

	4 Clusters	8 Clusters	16 Clusters	32 Clusters	64 Clusters
Running Time	13.077	39.892	75.575	147.198	526.874
Error Rate	1270476.0	1266416.2	1260343.6	1246426.8	1224471.0
Standard Deviation	4040.58	3068.42	2241.46	1573.78	1047.57

Topic Entropy	3.927	3.960	3.932	3.973	3.915

Table 3: Results of K-means clustering using Euclidean distance on documents that had valid classes.

Note on error rate: The error rate appears to be extremely high but this is only because the the square root was not taken of the euclidean distances. This was done simply save on computational power.

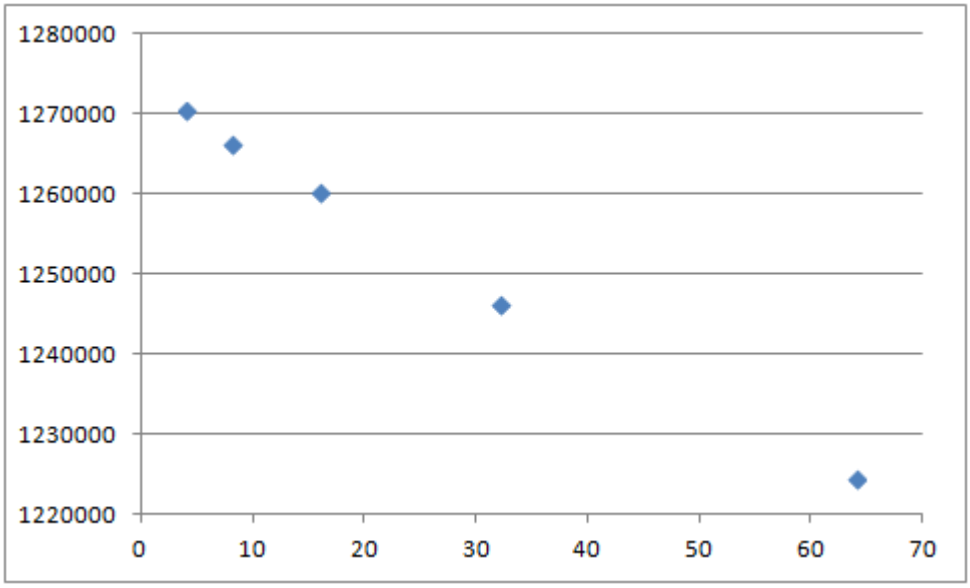

Figure 2: Error rate plot of K-means clustering using Euclidean distance on documents that had valid classes.

	4 Clusters	8 Clusters	16 Clusters	32 Clusters	64 Clusters
Running Time	63.092	105.879	118.755	690.134	2145.731
Error Rate	2554562.0	2549633.8	2542671.0	2522868.0	2497810.0
Standard Deviation	7826.128844294604	5839.341774496078	4361.125298715201	3070.157927883912	2134.5391911018196
Topic Entropy	7.73705126931187	7.769988067781509	7.732720996073548	7.699845117691675	7.735192498081731

Table 4: Results of K-means clustering using Euclidean distance on all valid documents.

Hierarchical Clustering and Manhattan Distance

Regarding this algorithm, not only that it was much slower, we measured the power consumption of this run - it used 8043 mAH (which means - almost a full battery charge of a laptop).

Running Times

The running time for hierarchical clustering is the same for all the cluster sizes.
The idea is that we build the structure once, and reuse it to calculate the entropies for the different hierarchies.
The running time for building the hierarchical cluster was : 8942.383 seconds

Results

	4 Clusters	8 Clusters	16 Clusters	32 Clusters	64 Clusters	128 Clusters
Entropy	1.6395	1.6386	1.6371	1.6334	1.6263	1.6114
Variance	4043.472610269544	3086.552395067999	2257.206981952475	1619.691848344539 3	1150.149853942275 5	810.7790522891297

Table 5: Results of hierarchical clustering using Manhattan distance on documents that had valid classes.

Hierarchical Clustering and Euclidean Distance

Running Times

The running time for hierarchical clustering is the same for all the cluster sizes.
The idea is that we build the structure once, and reuse it to calculate the entropies for the different hierarchies.
The time is 13946.535

Results

	4 Clusters	8 Clusters	16 Clusters	32 Clusters	64 Clusters	128 clusters
Entropy	1.6433	1.6403	1.6396	1.6366	1.6293	1.6196
Variance	4043.472610269544	3086.930342184611	2256.948794141107 6	1618.434659198124 8	1145.992459736728 5	806.7861975181897

Table 6: Results of hierarchical clustering using Euclidean distance on documents that had valid classes.

Summary

These are all the variances and entropies plotted.

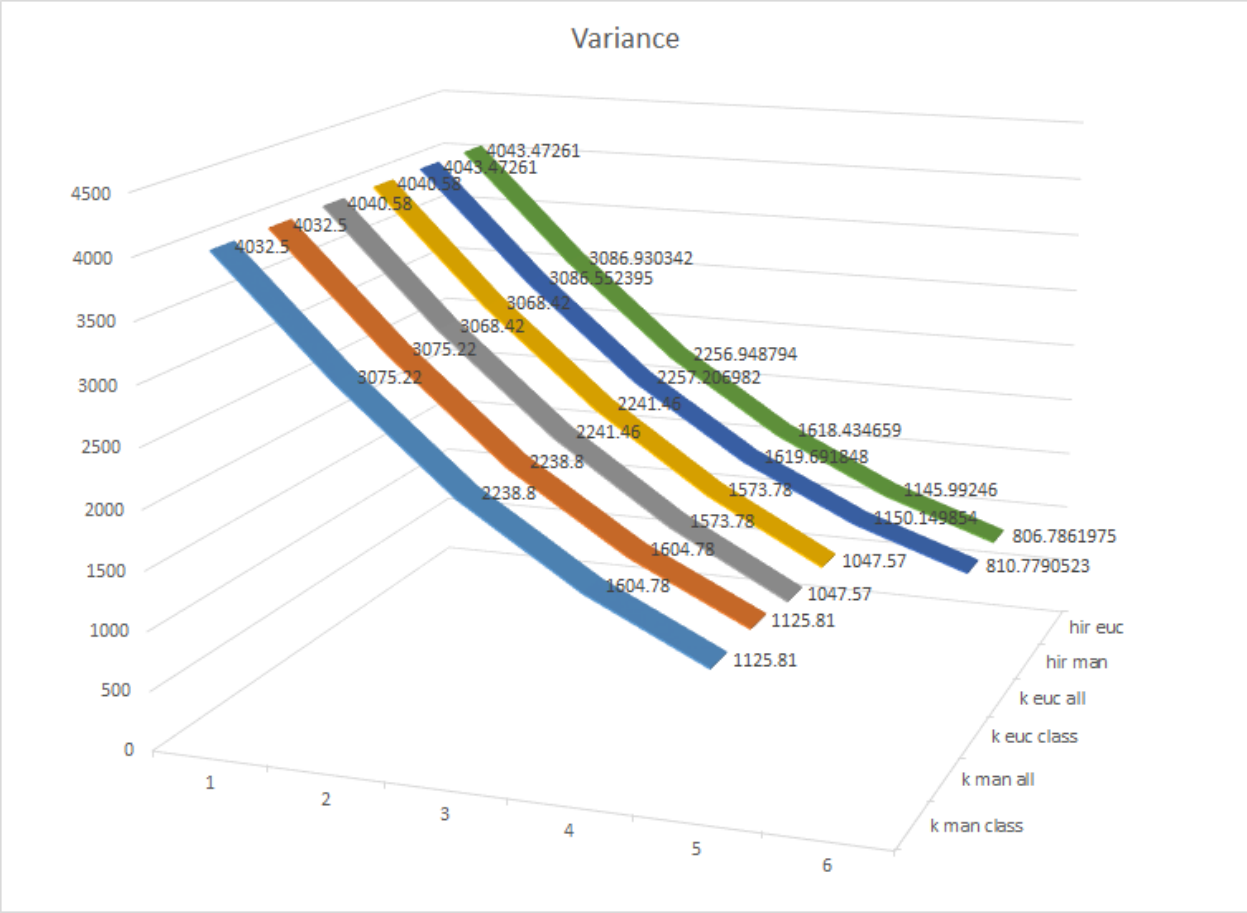

Figure 3: Variance plotted over number of clusters over type of clusterings.

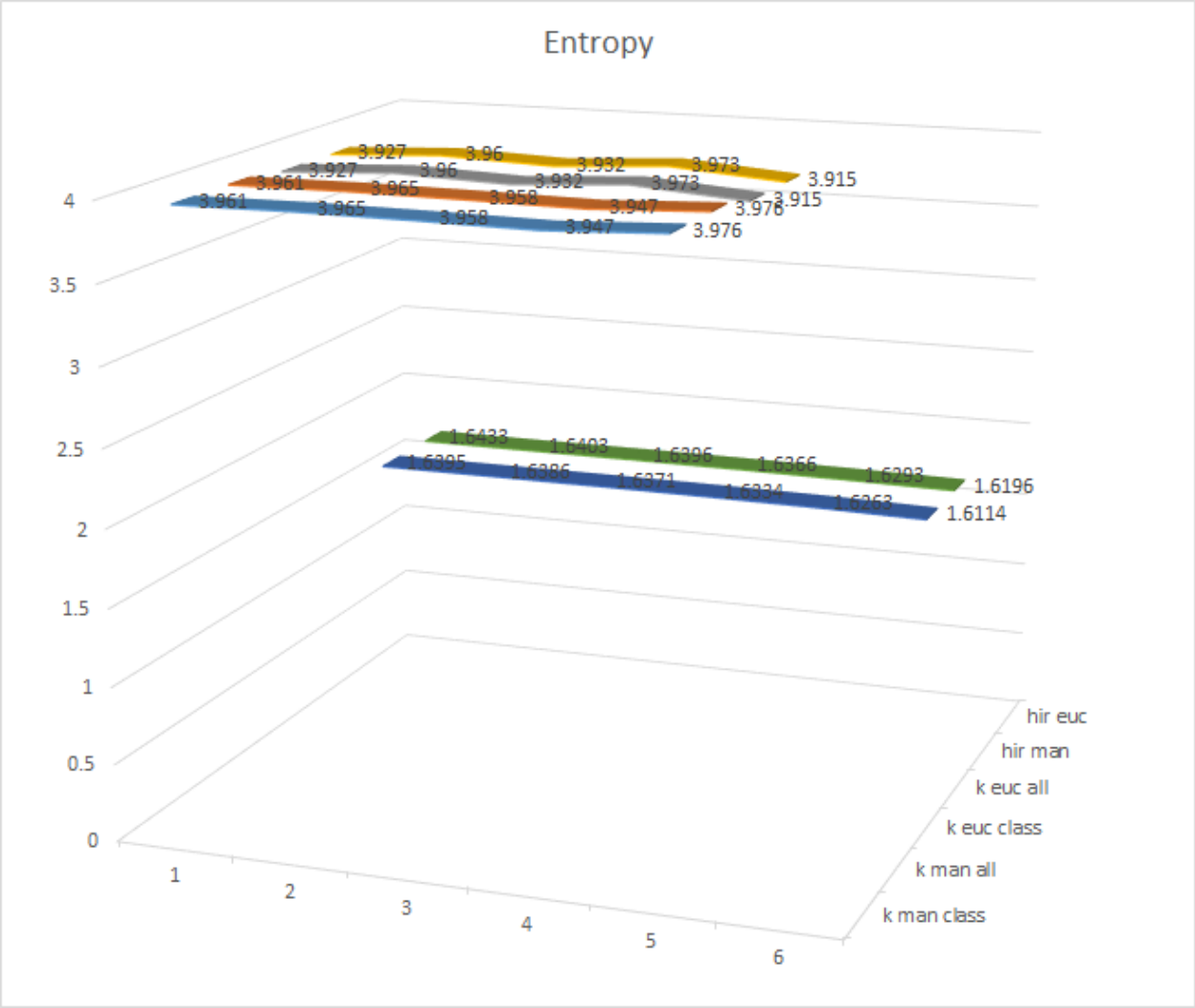

Figure 4: Entropy plotted over number of clusters over type of clusterings.