

## Part 4

Let,  $N$  = the number of unique words in the text file &  $S$  = the maximum number of unique words any word appears with

a)

Q: What data structure did you finally use for vectors?

HashMap where key is String, the unique words in the text file, and value is another HashMap that represents semantic descriptor vector for each unique word.

HashMap<String, HashMap<String, Double>>

Q: What is the asymptotic memory usage of a vector?

Single vector is a HashMap where the key is the unique words that appears with, and value the value is the distance. Therefore, the asymptotic memory usage of a vector is  $O(S)$ .

Q: Of all of your vectors?

As mentioned, data structure for vectors is HashMap where the keys are all the unique words in the text file, and values are semantic vectors of each unique word. Therefore, asymptotic memory usage of all vectors is  $O(N * S) = O(NS)$ .

Q: Is the memory usage reasonable and why?

$O(NS)$  is reasonable. In order to calculate the similarities, regardless of the measuring methods, single semantic vector has to have all distance values with given word. Therefore, memory usage of a vector should at least be  $O(S)$ . Also, all vectors mean the semantic vectors for all the unique words in the text file. Therefore,  $O(NS)$  is reasonable.

b)

Q: What algorithm did you finally use for cosine similarity?

Assume we are calculating cosine similarity of Word1 and Word2.

Since cosine similarity multiplies similarity value of each word in the denominator, we can ignore any zero similarities. Therefore, for the denominator, find the word that is contained in both semantic vectors of Word1 and Word2.

For the numerator, simply sum all the squared value of similarities of both semantic values.

Q: What is its asymptotic running time?

Calculating cosine similarity takes three separate for loops. First, it iterates through the semantic vector of Word1 or Word2 (whatever which is shorter), which is  $S$ . Also, second and third for loops iterates through semantic vector of Word1 and Word2 alternatively. Therefore, asymptotic running time is  $O(S+S+S) = O(S)$

Q: Is this running time reasonable and why?

$O(S)$  is reasonable. In order to calculate the similarity, it has to iterate through the semantic vector of particular word anyway. Therefore,  $O(S)$  is reasonable.

c)

Q: What algorithm did you finally use for the Top-J calculation?

PriorityQueue that takes Entry<String, Double> class as its data. Priority is defined with Double value that represents distance inside the Entry.

Q: What is its asymptotic running time (might be in terms of J, too)?

Top-J calculation takes two separate for loops. One has N iteration, and the other one has J iteration. Therefore, asymptotic running time is  $O(N+J) = O(N)$ .

To be specific, since cosine similarity should be calculated while constructing Top-J query, asymptotic running time will be  $O(N * S) = O(NS)$

Q: Is this running time reasonable and why?

Excluding cosine similarity calculation,  $O(N)$  is reasonable. Since Top-J calculation also prints out the words that have zero similarity with give word Q, it has to iterate through all unique words N. Therefore,  $O(N)$  is reasonable.

d)

Q: What improvements did you make from your original code to make it run faster?

We changed data structure for vector from two ArrayLists to HashMap. Originally, it was combination of ArrayList<String> that stores words and ArrayList<Double>. However, it took long time to traverse through the list. So, we changed the data structure to HashMap<String, Double>, and the time spent became lot faster.







Q: Give an example of your running time measurements before and after the changes. Describe the information that informed your choices (asymptotic running time analysis, asymptotic memory analysis, and/or profiling).

When the data structure for vector was two ArrayLists, using K and W notation (K = the number of sentences, and W = the maximum number of words in the sentence), asymptotic running time of all vectors was  $O(K * W * (S + (W + S)))$ . However, it became lot faster to  $O(KW^2)$  when the data structure for vector was changed to HashMap.

## Part 5

Q: Does the Top-J get better with more data? Index one of the books. Then, run an interesting Top-J query. Then, in the same session, index an additional book (at this point your vectors reflect both books) and run the Top-J again. Report the results and comment on whether they changed.

(Larger size texts are indexed in order: 89BK, 1096KB, 3281KB)

|   |   |   |          |        |
|---|---|---|----------|--------|
|  |  | edu.uiowa.cs.similarity.TopJCommand.topj (java.util.HashSet, java.util.He | 41.0 ms  | (0.1%) |
|  |  | edu.uiowa.cs.similarity.TopJCommand.topj (java.util.HashSet, java.util.He | 157 ms   | (0.1%) |
|  |  | edu.uiowa.cs.similarity.TopJCommand.topj (java.util.HashSet, java.util.He | 1,088 ms | (0.5%) |

Top-J takes longer time as another text is indexed. Since Integer J in Top-J calculation was fixed as 10, it seems like there are more unique words in the longer text.

Q: How does Top-J change with different measures? Run your Top-J query above for all three similarity measures and report your findings. Discuss what is the same and different about the results for the 3 similarity measures.

Larger size texts are indexed in order: 89BK, 1096KB, 3281KB

Cosine similarity: 31.3ms / 152ms / 998ms

Euclidean distance: 62.4ms / 375ms / 3391ms

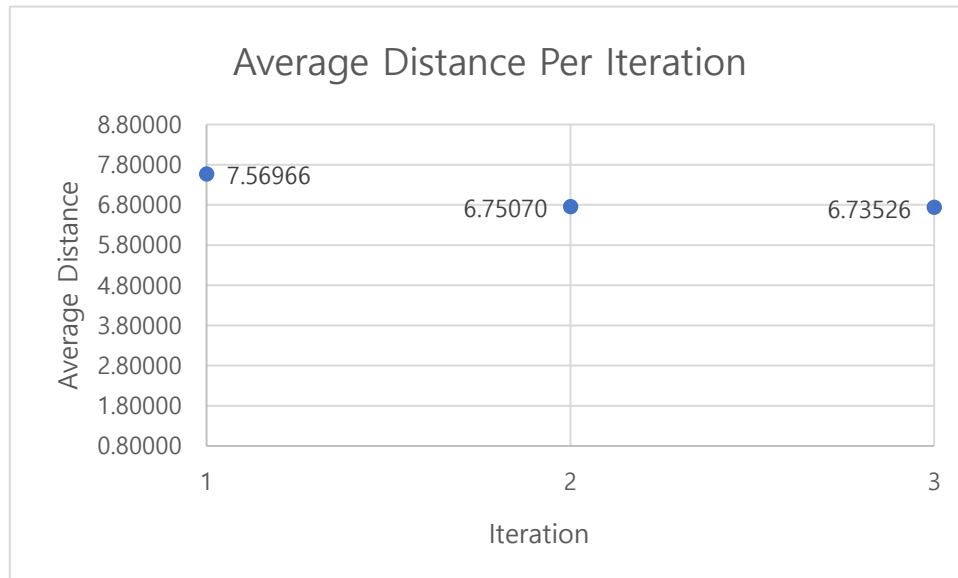
Euclidean distance with normalized vectors: 93.9ms / 609ms / 5179ms

In common, about 6~7 is multiplied from first Top-J to second, and about 8~9 is multiplied from second Top-J to third. Since same files are indexed for each similarity measure, similar number of multiplier for each step is very reasonable

In contrast, since their asymptotic runtimes are different, their actual runtime varies. That is, using the asymptotic runtime notation, cosine is  $3 * O(S)$ , Euclidean is  $4 * O(S)$ , and Euclidean with normalized vectors is  $6 * O(S)$ .

## Part 6

Q: Using the output, create a scatterplot (using any program you like, e.g., Excel) showing the average distance cs. iteration. After how many iterations did the clustering seem to converge (i.e., not change anymore)?



Clustering is assumed to be converged and iteration soon breaks when difference between previous average distance and current average distance is less than 0.00001. In this case, k-means clustering was experimented with  $k=3$  clusters for 100 iteration, and clustering is converged at 3<sup>th</sup> iteration. That is, there is no significance difference in centroids between 3<sup>th</sup> and 4<sup>th</sup> iteration.

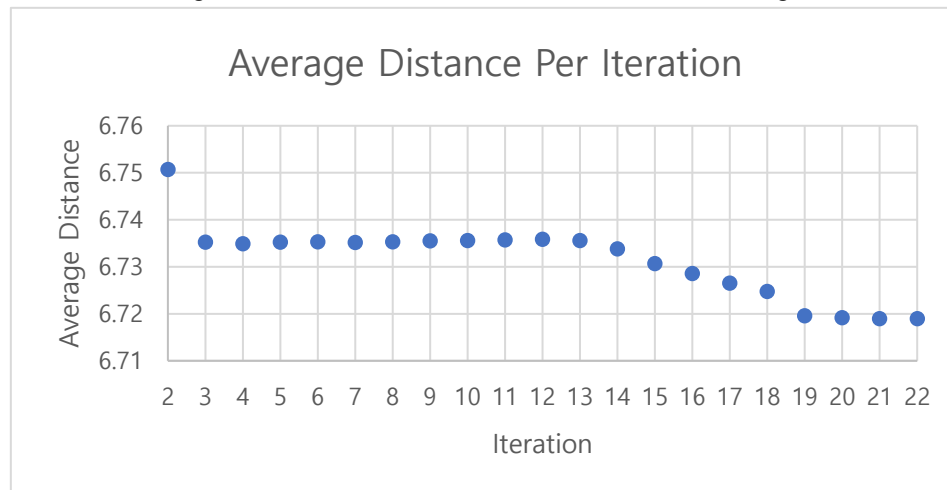
## Extra credit 1

Q: Run k-means clustering on one or more of the larger texts. Run it once for at least 2 different values of k. For each value of k, create a scatterplot as above. Pick the number of iterations to be big enough that the algorithm converges. Write a description of your experiment and explain the results.

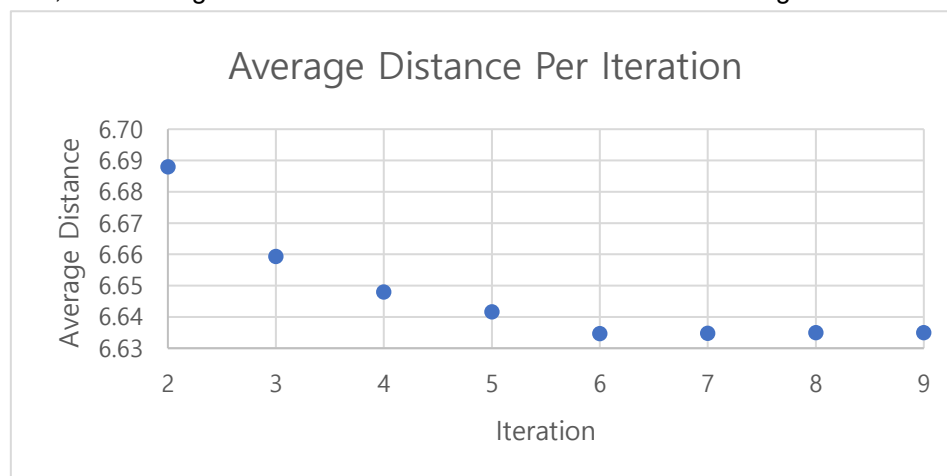
89KB *War and Peace* text file is indexed for the experiment

In order to clearly see whether convergence is made properly, "Average Distance Per Iteration" plots below excluded 1<sup>st</sup> iteration.

K = 2, final average distance = 6.71892650725799. Cluster converges at 22<sup>nd</sup> iteration.



K = 3, final average distance = 6.63493820819507. Cluster converges at 9<sup>th</sup> iteration.



K = 4, final average distance = 6.58271518075897. Cluster converges at 8<sup>th</sup> iteration.

K = 5, final average distance = 6.51272966577977. Cluster converges at 16<sup>th</sup> iteration.

...

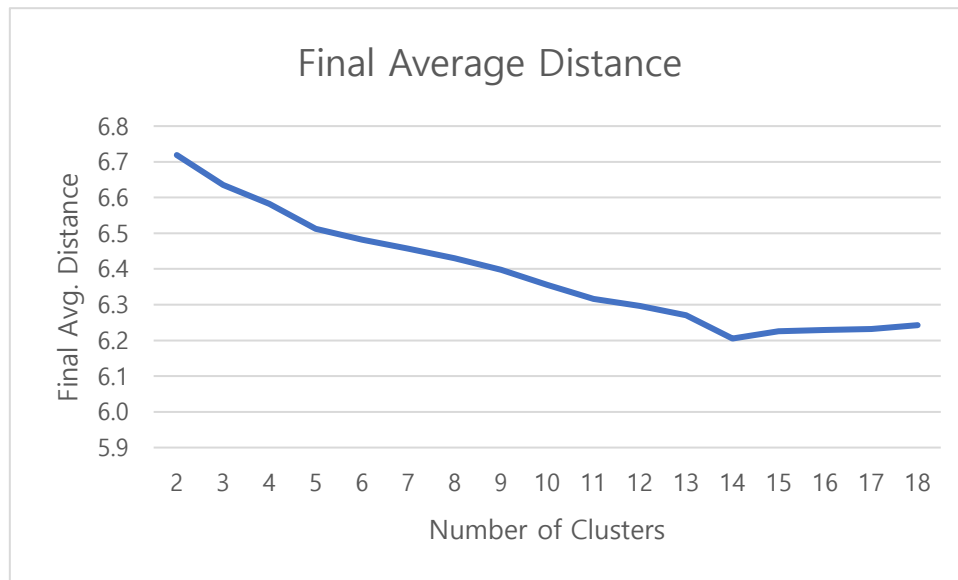
K = 17, final average distance = 6.23229706481916. Cluster converges at 14<sup>th</sup> iteration.

K = 18, final average distance = 6.24301554994665. Cluster converges at 24<sup>th</sup> iteration.

Experiment was done for 17 times, from K = 2 to K = 18. We concluded that the number of

iteration take until the cluster converge is random, because of randomization of the first K number of initial points.

Also, according to the “Final Average Distance” plot below, final average distance is decreasing until K = 14, and tends to increase from K = 15. Therefore, 14 number of clusters best describes k-mean clustering of similarities between unique words in *War and Peace* text file



## Extra credit 2

Q: Run the k-means with representative words on one or more of the larger texts. Pick J between 2-4. Pick a reasonable k. Include a copy of your results and write about the interesting observations you find

(Clusters in the result is omitted from the second line, if it's longer than 2 lines.)

Result:

run:

> index War and Peace (89KB).txt

Indexing War and Peace (89KB).txt

> kmeans 14 100

Iteration: 1 / Avg. Distance: 7.106269082291377

Iteration: 2 / Avg. Distance: 6.481401150579967

Iteration: 3 / Avg. Distance: 6.399683469250953

Iteration: 4 / Avg. Distance: 6.337020455736456

Iteration: 5 / Avg. Distance: 6.30094865670012

Iteration: 6 / Avg. Distance: 6.292725563682442

Iteration: 7 / Avg. Distance: 6.289922502029991

Iteration: 8 / Avg. Distance: 6.288327711541457

Iteration: 9 / Avg. Distance: 6.286222617637704

Iteration: 10 / Avg. Distance: 6.284990753896754

Iteration: 11 / Avg. Distance: 6.284018004109822

Iteration: 12 / Avg. Distance: 6.282615830660741

Iteration: 13 / Avg. Distance: 6.281118809495672

Iteration: 14 / Avg. Distance: 6.280504825328594

Iteration: 15 / Avg. Distance: 6.2808488190767715

Iteration: 16 / Avg. Distance: 6.280746419590178

Iteration: 17 / Avg. Distance: 6.280728581140465

Clustering is converged at iteration 17

Cluster 0

[platter, german, govern, hall, embosom, pick, warrant, sake, prison, sister, pretens, ...]

Cluster 1

[inquir, infinit, alleg, run, hurri, float, quest, put, concern, lip, find, pleasuresprovid, collat, ...]

Cluster 2

[gun, sabbath, broke, reverber, natur, echo, around, wander, prolong, quiet, angri, roar, peculiarli, noontim]

Cluster 3

[half, year, maintain, intellig, rider, bank, bar, poor, farm, partli, ten, ", ran, ghostli, serv, ...]

Cluster 4

[door, wool, spun, glisten, egg, mingl, corn, shovel, bag, stood, festoon, knowingli, red, loom, ...]

Cluster 5

[practic, someth, draw, joke, disposit, boorish, plai, rustic, system, extrem, fund, waggeri, altern, pacif, provok, obstin, rival]

Cluster 6

[(doe, reason, inform, depend, (avail, addit, (or, re-us, without, (a), permiss, "plain, damag, ...]

Cluster 7

[ichabod, whole, tassel, upon, scene, countri, school, sever, road, fear, brook, still, van, pass, ...]

Cluster 8

[splutter, sourc, thought, told, haunt, horseman, tale, dutch, satan, path, fire, despit, terror, ...]

Cluster 9

[round-crown, rope, jacket, back, manag, mount, half-broken, halter, negro, appear, fragment, tow-cloth, trowser, mercuri, interrupt, rag, hat, suddenli, wild, wai]

Cluster 10

[date, septemb, 25, legend, languag, 2008, releas, post, [ebook, 1992, english, titl, 14, last, ...]

Cluster 11

[stripl, feather, red-tipt, splendid, scream, pretend, good, cloud, twitter, jai, white, broad, ...]

Cluster 12

[kettl, beheld, wagon, busi, lord, realiz, hope, famili, kentucki, katrina, pot, bloom, load, heel, ...]

Cluster 13

[brom, cossack, don, till, startl, troop, listen, moment, crew, sleep, halloo, sometim, whoop, midnight, gang, clatter, past, exclaim, hurry-scurri, bone, farmhous, dame, goe, "ai, dash]

> representatives 3

3 representative(s) from Cluster 0

[d, heaven, indol]

3 representative(s) from Cluster 1

[inquir, infinit, lower]

3 representative(s) from Cluster 2

[sabbath, reverber, wander]

3 representative(s) from Cluster 3

[intellig, ten, aliv]

3 representative(s) from Cluster 4

[spun, glisten, dri]

3 representative(s) from Cluster 5

[disposit, system, obstin]

3 representative(s) from Cluster 6

[parti, 20%, calcul]

3 representative(s) from Cluster 7

[behind, scene, heard]

3 representative(s) from Cluster 8

[satan, wasa, phantom]

3 representative(s) from Cluster 9

[round-crown, rope, halter]

3 representative(s) from Cluster 10

[septemb, [ebook, 1992]

3 representative(s) from Cluster 11

[feather, splendid, golden-wing]

3 representative(s) from Cluster 12

[kettl, kentucki, nai]

3 representative(s) from Cluster 13

[cossack, till, crew]

>

From the Extra Credit 1,  $K = 14$  of clusters seem to converge the words in *War and Peace* text file better than other number of clusters. Therefore,  $K = 14$  is picked.

Obviously, none of representative words duplicate.

Maybe we just don't understand the word since it's stemmed, but unlike what we expected, 3 representative words in most cluster do not seem to have direct relation.