# CSE494 Information Retrieval
# Project C Report

**By: Jianchun Fan**

## Introduction

In project C we implement several different clustering methods on the query results given by pageRank algorithms. The clustering methods implemented are: k-means algorithm and two of the extended k-means algorithms (asynchronous update of the centroids and the multiple k-means method), Hierarchical agglomerative clustering algorithm, bisecting k-means algorithm and buckshot clustering algorithm.

In this report, each of the above algorithm and the implementation is discussed. Also we will analyze and compare the performance of the different algorithms. Then some of the observations during the implementation are analyzed. At last the GUI implemented is briefly introduced. The source code is attached and some of the testing results are included.

## Algorithms and implementation

### 1. k-Means

#### Algorithm

The basic idea of k-Means algorithm is to do a local optimization on a given number of clusters. Specifically, first we need to randomly pick up k documents from the entire collection and make them as the initial centroids of the desired k clusters. Then for each document in the collection find the nearest centroid and put this document into the corresponding cluster. After each document is assigned to one of the cluster, recompute the centroids and repeat the computation. This method iteratively optimize the clusters until the computation converge when the clusters do not change anymore and the clustering quality achieved a local maximum. The advantage of the k-means is that its complexity is very low and is very easy to implementation. But it also has the problem of instability: the final result will change dramatically if the initial centroids change, and it is very sensitive to the outliers because the way of its clustering is exhaustive.

#### Implementation

The implementation of the k-means algorithm is very straightforward. First we represent the entire collection of the documents as a vector of term vectors, and then randomly pick up k of the documents to be the initial seeds of the k clusters. A predicate function is defined to check whether the computation is converged at the end of each iteration.

#### Performance

During the testing we find that k-means works reasonably well given its low cost. The quality of the clustering seems to be influenced by the initial collection of the documents. Although it does not always give very good clusters, for some particular document sets, the clustering more or less reflect some of the natural clustering. For example for the page collection returned by pageRank algorithm for query “Computer Science”, if we want to have 3 clusters, k-means method usually give the 3 clusters similar to the following: one of them is more relevant to “women in computer science organization”; another of them is more about the general information of computer science department and yet another is more likely to be about the computer science courses.

And we also noticed that the k-means is very sensitive to the choice of the initial seeds. In the implementation the seeds are randomly picked and we found that when we run k-means on the same collection for multiple times the result changes dramatically sometimes.

## 2. extended K-Means

### Algorithm

The extended K-Means algorithm has the same basic idea of the k-means method. The only difference is that instead of update the centroids of the clusters at the end of each iteration, we update them more frequently. Specifically, each time after several documents are assigned to the clusters, we immediately recomputed the centroids according to the assigned documents and use them to cluster the rest documents.

### Implementation

The implementation is straightforward. We just need to have a counter to count the number of changes we’ve made and update the centroids each time the counter gets to some small integer.

### Performance

As for the quality of the clustering, according to the testing, it seems that this kind of extension of k-means method does not noticeably improve the quality. But we can notice that it will make the convergence of the computation faster. In the experiments, it almost always needs less iteration to converge. For example for the given query “Computer Science” and number of clusters as 3, it usually takes regular k-Means 7~9 iterations to converge, but it only take 3 or 4 iterations for the extended k-Means to converge.

## 3. Multiple k-Means

### Algorithm

This is another extension of the regular k-means method. The basic idea is just to run the k-means for several times and pick the best clustering we get as the final result.

### Implementation

The implementation only has one issue to be solved, which is how to find the “best” clustering got from the multiple running of k-Means method on a same collection of documents. In my implementation we use the average pair-wise similarity in a cluster as the measure of the quality of the cluster, and use the average quality measure of all the clusters as the measure of the quality of the entire clustering. Then we just run k-means several times using different randomly picked seeds and keep the clustering with the best quality measure as the final results.

A very small issue in the implementation is that the average similarity of a cluster is just the square of the magnitude of the centroid. But this holds only if all the documents are originally normalized to be unit vectors, otherwise we can not directly use this fact to simplify the computation.

**Performance**

The quality of the clustering got by multiple k-Means is more satisfactory. Actually during the testing we noticed that the multiple k-Means method almost always out performs the regular k-means. The reason is obvious. A more interesting issue is to find out what is a better way to measure the quality of the clustering, specifically, how to combine inter-cluster quality measure and intro-cluster quality measure together.

## 4. HAC(Hierarchical Agglomerative  Cluster)

### Algorithm

The basic idea of HAC if to start from many clusters and keep merging the closest cluster pairs until we get desired number of clusters. The result of the clustering will be a hierarchical structure. To do this the important issue is to find a way to measure the distance between two clusters.

### Implementation

The implementation of the HAC is, first we put every document in the collection into a singleton cluster which contains just one document. Then we keep finding the closest cluster and merge them, and use the merged one as a new cluster. To do this a merge function is implemented, and a function to measure the distance of two clusters is implemented. In my implementation, the group average distance is used, which is the average distance of all pairs of documents with one from each cluster.

### Performance

The quality of clustering got from HAC is satisfactory. It is usually better then regular k-means. But the computation of HAC is costly and if the initial document collection is large the computation could be very time consuming. In the experiment if we have 100 documents as the initial collection, the computation will take a couple of minutes, much longer than other algorithms.

## 5. buckshot Algorithm

### Algorithm

The buckshot algorithm tries to achieve good quality of clustering as HAC without as much cost. It is actually a hybrid method using the features of both HAC and k-Means. Basically it first randomly pick sqrt(n) documents from the entire collection and run HAC on them. Then use the results as the initial seeds for the k-Means algorithm. This method avoids using HAC on the entire set, and also finds possibly better seeds for the k-Means.

### Implementation

The implementation of this algorithm is very simple, given the fact that we've already implement k-Means and HAC.

### Performance

In the testing we found that the quality of the clustering is good, compared to k-means and extened k-means. Also the time consumed is much less that the HAC. But we also noticed that this method is also not so stable: if we run it several times, for some document collection the result could be quite different. The reason is that although the HAC stage might find better seeds for k-Means, the subset to be used in HAC is still randomly chosen. And in out experiment setting the size of the initial collection is small which makes it more sensitive to the randomness.

## 6. Bisecting k-Means

### Algorithm

The bisecting k-Means looks like the opposite of HAC, because it starts from just one cluster with all the documents inside, ant then in each iteration, find one of the leaf clusters and split it into two smaller clusters. This keeps going until the desired number of clusters is achieved. To find the leaf cluster to be split, it could be as simple as just finding the largest one, or to get better results, finding the cluster with worst cluster quality. To split the cluster, we can run k-means on it for multiple times and use the best result among them.

### Implementation

The implementation of the bisecting k-Means is straightforward, because we already have the cluster quality measure function to find the leaf cluster to be split, and we also have implemented multiple k-Means which is used to actually split the selected leaf nodes.

### Performance

The Bisecting k-Means method has low cost, compared to HAC, yet its quality of clustering is very satisfactory. In the experiment it almost always gets at least as good quality as HAC, and it is also more stable than buckshot in our setting. The reason is that during the split, we used multiple k-means which tends to find a better clustering among several candidate and this reduce the effect of the randomness. In my experiment this method seems to be more stable than buchshot method, i.e. the result of the clustering does not change dramatically when we run it on same document set for several times.

## Important observations

### The effect of changing the number of clusters

During the experiment we noticed that when we increase the number of the clusters, the result of the clustering tends to less likely to be corresponding to any natural clustering. One reason is that the documents to be clustered are returned by the pageRank method which combines the document vector similarity and pageRank value. Thus unlike a random collection of documents, these documents are naturally sharing more common features and are likely to be close to each other; Another reason is that the size of the collection we used in the clustering is small, thus if there are too many clusters the documents are intentionally separated although they actually are not so far away from each other, and as a result, this kind of clustering is more sensitive to those random factors and is less likely to reflect any natural clusters.

## The effect of changing the size of the collection

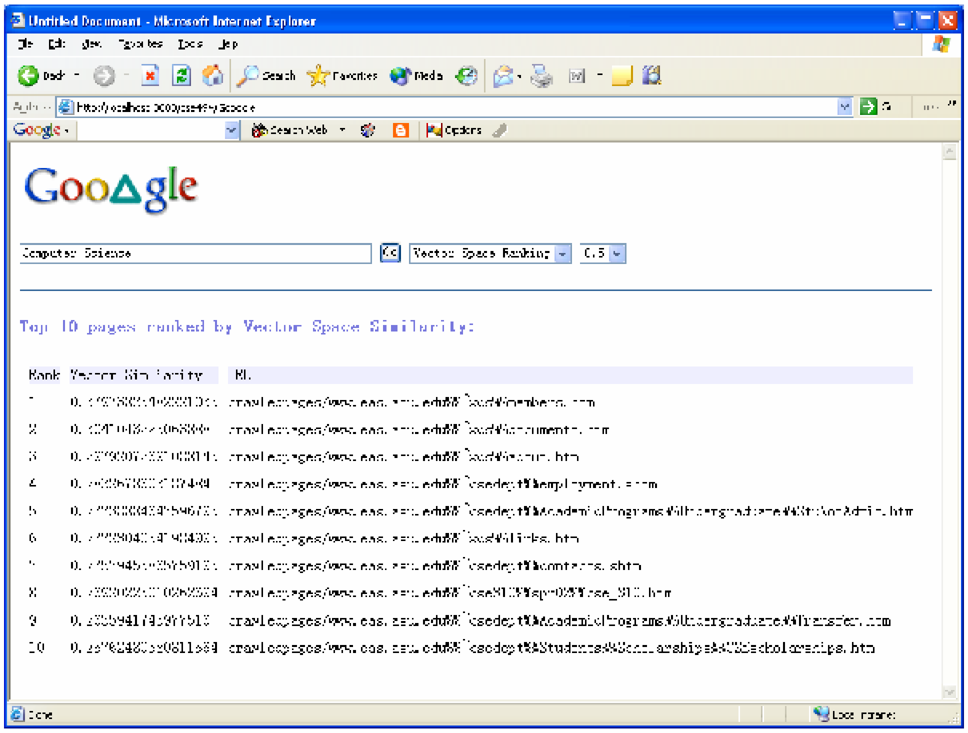
During the experiment several different sizes of the collection (50, 70, 100) are tested and we found that for a fixed number of clusters, when the size of the collection is larger, the quality of the clustering is usually better. The possible reason is that in our setting we only show top 3 documents of each of the cluster instead of the entire cluster, and it is actually straightforward that it is more likely to find documents close to each other in a large set other than a small set. In this case, although the overall quality of each cluster (for example, average similarity from the centroid) might not be better, for the top 3 documents, it will be easier to find some of them very close to each other given the larger document collection.

## The implementation of the GUI

The GUI used in the project is implemented by using Java Servlet. It has very simple interface for user to specify query, and choose the ranking method to be used (Vector Similarity, A/H Ranking, or pageRank). Also the user can specify the weight of pageRank value if pagRank is used in ranking.

After the query result is shown to the user, if the user chose pageRank, then the GUI will allow the user to choose a clustering method (k-Means, extended k-Means, multiple k-Means, bisecting k-Means, HAC, buckshot) and specify the number of clusters desired. Then the GUI will show the clusters (only show the top 3 documents in each Cluster).

The following is the screen shot of the GUI:

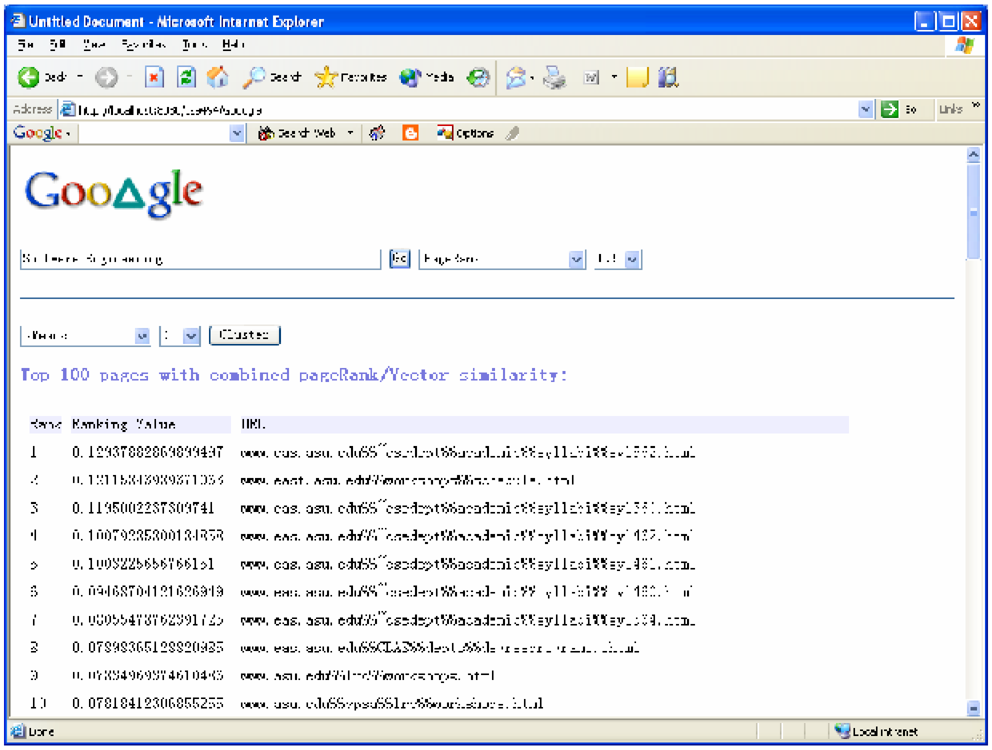


Computer Science

Vector Space Ranking 0.5

Top 10 pages ranked by Vector Space Similarity:

| Rank | Vector Similarity | URL |
|---|---|---|
| 1 | 0.[illegible] | crawledpages/www.eas.asu.edu%%~csd%%members.htm |
| 2 | 0.[illegible] | crawledpages/www.eas.asu.edu%%~csd%%documents.htm |
| 3 | 0.[illegible] | crawledpages/www.eas.asu.edu%%~csd%%about.htm |
| 4 | 0.[illegible] | crawledpages/www.eas.asu.edu%%~csedept%%employment.shtm |
| 5 | 0.[illegible] | crawledpages/www.eas.asu.edu%%~csedept%%AcademicPrograms%%Undergraduate%%StudentAdmin.htm |
| 6 | 0.[illegible] | crawledpages/www.eas.asu.edu%%~csd%%links.htm |
| 7 | 0.[illegible] | crawledpages/www.eas.asu.edu%%~csedept%%contacts.shtm |
| 8 | 0.[illegible] | crawledpages/www.eas.asu.edu%%~cse310%%spr03%%cse_310.htm |
| 9 | 0.[illegible] | crawledpages/www.eas.asu.edu%%~csedept%%AcademicPrograms%%Undergraduate%%Transfer.htm |
| 10 | 0.[illegible] | crawledpages/www.eas.asu.edu%%~csedept%%Students%%Scholarships%%Scholarships.htm |



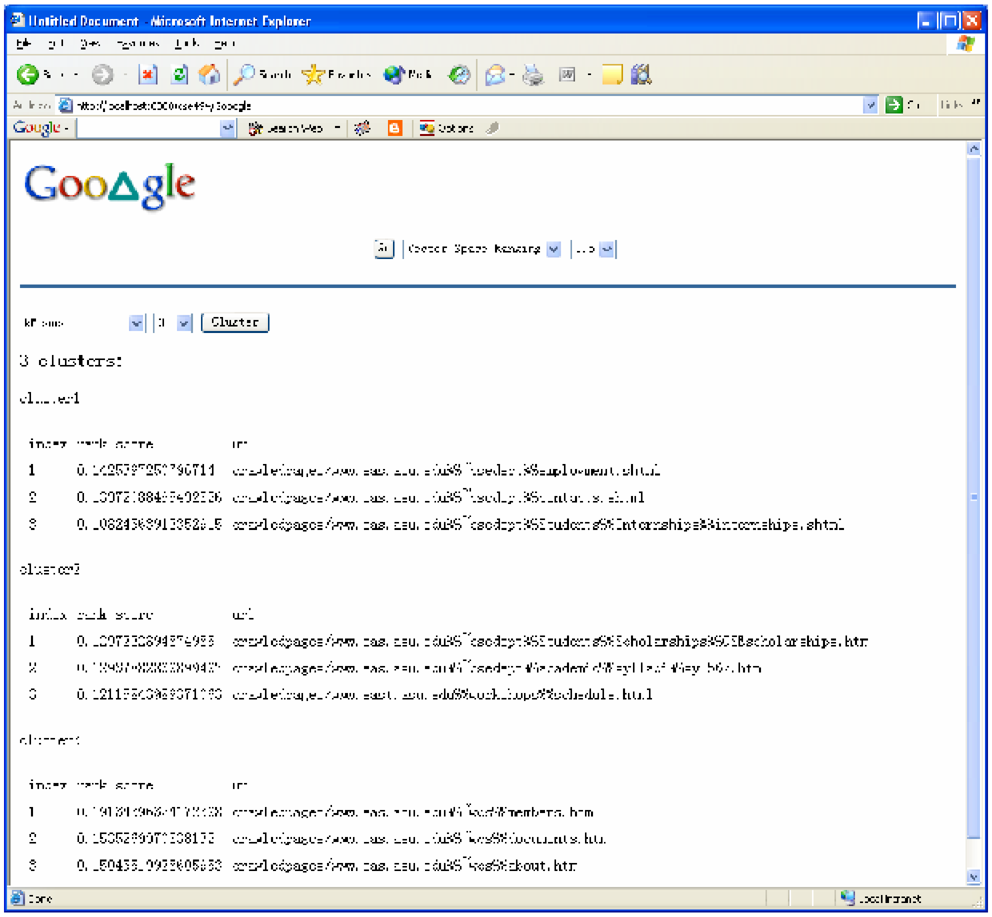
Software Engineering

PageRank [illegible]

Cluster

Top 100 pages with combined pageRank/Vector similarity:

| Rank | Ranking Value | URL |
|---|---|---|
| 1 | 0.12937882869899437 | www.eas.asu.edu%%~csedept%%academic%%syllabi%%syl[illegible].html |
| 2 | 0.1211[illegible] | www.east.asu.edu%%[illegible].html |
| 3 | 0.11950022237309741 | www.eas.asu.edu%%~csedept%%academic%%syllabi%%syl[illegible].html |
| 4 | 0.1007[illegible] | www.eas.asu.edu%%~csedept%%academic%%syllabi%%syl432.html |
| 5 | 0.1009[illegible] | www.eas.asu.edu%%~csedept%%academic%%syllabi%%syl481.html |
| 6 | 0.0946870412162691[illegible] | www.eas.asu.edu%%~csedept%%academic%%syllabi%%syl[illegible].html |
| 7 | 0.0805[illegible] | www.eas.asu.edu%%~csedept%%academic%%syllabi%%syl[illegible].html |
| 8 | 0.07898365128820935 | www.eas.asu.edu%%CLAS%%[illegible].html |
| 9 | 0.0789[illegible] | www.asu.edu%%[illegible].html |
| 10 | 0.07818412306855255 | www.asu.edu%%[illegible].html |



```
3 clusters:

cluster1

 index rank score           url
 1     0.14253972527960711  crawledpages/www.eas.asu.edu%5~csedept%5employment.shtml
 2     0.13072188497402136  crawledpages/www.eas.asu.edu%5~csedept%5contacts.shtml
 3     0.10824369913352615  crawledpages/www.eas.asu.edu%5~csedept%5Students%5Internships%5internships.shtml

cluster2

 index rank score           url
 1     0.12973326949749833  crawledpages/www.eas.asu.edu%5~csedept%5Students%5Scholarships%5CSEscholarships.htm
 2     0.12937[illegible]   crawledpages/www.eas.asu.edu%5~csedept%5Academic%5Syllabi%5Fey 567.htm
 3     0.12115[illegible]   crawledpages/www.east.asu.edu%5workshops%5schedule.html

cluster3

 index rank score           url
 1     0.1918[illegible]    crawledpages/www.eas.asu.edu%5~esd%5members.htm
 2     0.15052990771308173  crawledpages/www.eas.asu.edu%5~esd%5documents.htm
 3     0.15043510922605653  crawledpages/www.eas.asu.edu%5~esd%5about.htm
```