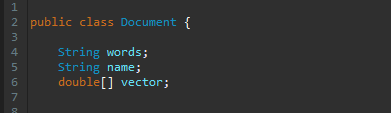
**CPS 842 Project #3 Report**

**System Design**

I created a document class to store the documents. Each document object would store the content of the document, the name of the document and the vector.



I created a centroid class which would be used to form the clusters. Each centroid would contain a list of documents in that cluster.

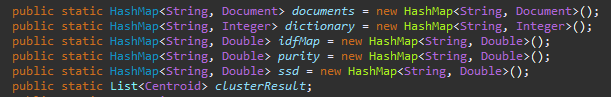


I would read all the files into document objects and store it into a hashmap, with the document name being the key and the value being the document object.

I would create a dictionary that would be stored in a hashmap as well. The dictionary would have every word as the key and their document frequency as the value.

Using the dictionary object, I can calculate the idf values for all the words and store them into a similar hashmap to dictionary except the value is the idf value.

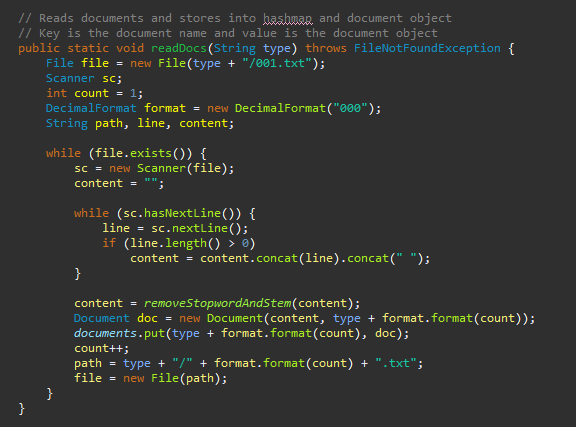
I would use a list of centroid objects to store the result of k-means clustering, and hashmaps to store the purity and ssd values of each cluster.



**Methods**

* readDocs(String type)

This method reads all the files inside a folder “type” and applies stopwords and stemming to the contents of the files and stores it into the documents hashmap.

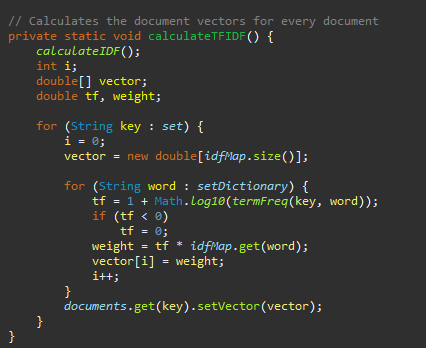


* readDictionary()

This method will read the dictionary file and store its contents into a hashmap with the word being the key and the value being the document frequency. (Only if dictionary created beforehand, else use createDictionary and write method)

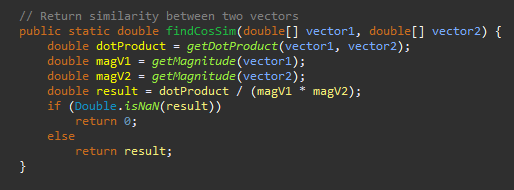
* calculateTFIDF()

This method calculates the document vector for all documents and stores the vectors to the corresponding document in the documents hashmap.



* findCosSim(double[] vector1, double[] vector2)

This method returns the similarity score between two vectors.



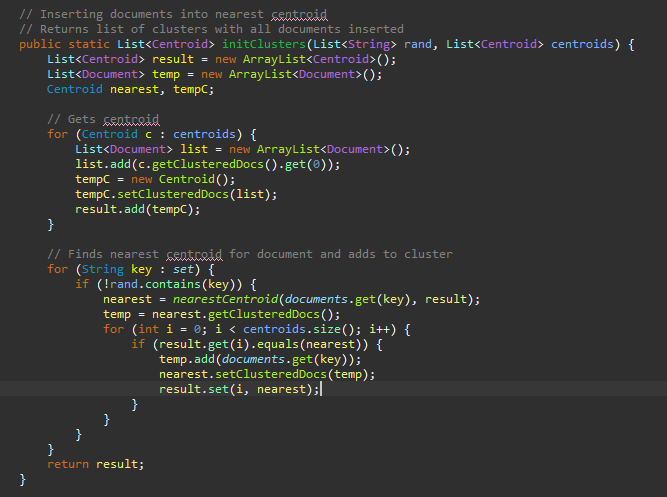
* kMeansCluster(int k)

This method returns a list of centroids with k clusters. It performs K-means clustering and stops after the centroids do not change or a certain amount of iterations.



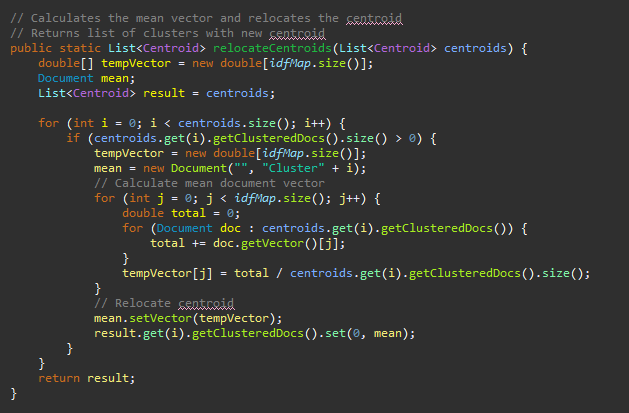
* initClusters(List<String> rand, List<Centroid> centroids)

This method returns a list of centroids after adding to each centroid, documents that are nearest to it. It uses cosine similarity to find the most similar centroid to the document and then adds that document to the centroid.



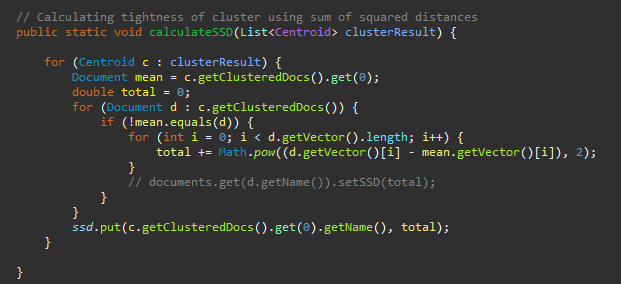
* relocateCentroids(List<Centroid> centroids)

This method returns a list of centroids with new mean centroids. It will calculate the mean document vector based on all documents inside the centroid and reassign the mean document vector to the centroid.



* calculateSSD(List<Centroid> clusterResult)

This method stores the sum of squared distances score for each cluster into the ssd hashmap. It gets the difference for each document vector and mean vector and adds it to the total sum.

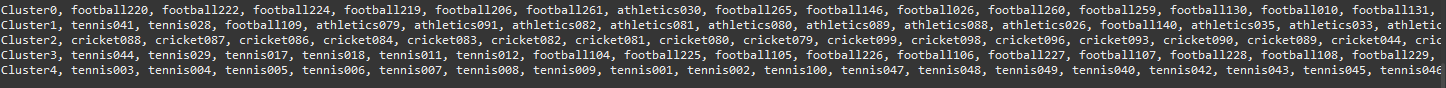


* calculatePurity(List<Centroid> clusterResult)

This method stores the purity score for each cluster into the purity hashmap. It finds the most common label inside the cluster and calculates a purity score based on how much of those documents with the same label is in the cluster.



**Examples #1**



**Purity scores:**

Cluster0(rugby): 0.5963302752293578

Cluster1(cricket): 0.45263157894736844

Cluster2(cricket): 0.9871794871794872

Cluster3(football): 0.7090301003344481

Cluster4(tennis): 0.5341614906832298

**Sum of Squared Distances:**

Cluster0: 47960.635901367736

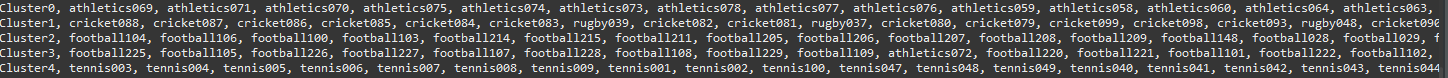
Cluster1: 27271.8137995188

Cluster2: 31316.445618029233

Cluster3: 80132.18861990413

Cluster4: 38442.97485964106

**Examples #2**



**Purity scores:**

Cluster0(athletics): 0.9438202247191011

Cluster1(cricket): 0.8333333333333334

Cluster2(football): 0.9888888888888889

Cluster3(football): 0.5

Cluster4(tennis): 0.7637795275590551

**Sum of Squared Distances:**

Cluster0: 23664.169043018668

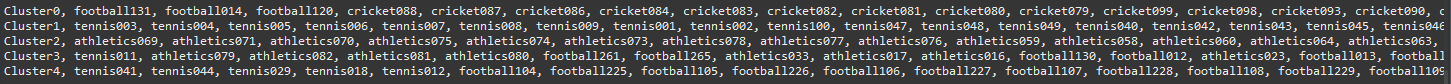
Cluster1: 42731.8135564249

Cluster2: 33199.60703219832

Cluster3: 95484.01054866353

Cluster4: 30670.230310080704

**Examples #3**



**Purity scores:**

Cluster0(cricket): 0.95

Cluster1(tennis): 0.989010989010989

Cluster2(athletics): 0.9659090909090909

Cluster3(rugby): 0.8

Cluster4(football): 0.6983240223463687

**Sum of Squared Distances:**

Cluster0: 31123.651469126064

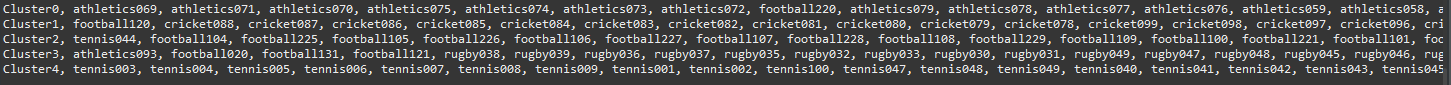
Cluster1: 20004.437066276692

Cluster2: 23672.75527861898

Cluster3: 42517.61078899143

Cluster4: 107849.45551897456

**Examples #4**



**Purity scores:**

Cluster0(athletics): 0.9433962264150944

Cluster1(cricket): 0.9827586206896551

Cluster2(football): 0.9391634980988594

Cluster3(rugby): 0.896774193548387

Cluster4(tennis): 0.9411764705882353

**Sum of Squared Distances:**

Cluster0: 28963.83635323

Cluster1: 42039.66362346912

Cluster2: 80994.56678761863

Cluster3: 50011.48730744872

Cluster4: 23128.59921353146