Data Glacier

Your Deep Learning Partner

Week 13 Deliverables

Name	Manoj Nagaraja
Email	manojn.7270@gmail.com
Country	United Kingdom
College	University of Liverpool
Specialization	Data Science
Github Repo Link	https://github.com/ManojN7270/Final-Project-week_7-to-week_13.git

k-means clustering algorithm

An unsupervised learning approach called K-means clustering is used to organise a collection of data points into groups or clusters. Partitioning the data into k different clusters, where k is a user-defined number. The process begins by choosing k initial centroids, or points that stand in for each cluster's center, at random. The closest centroid is then assigned to each data point. The algorithm calculates the mean of all the data points given to each centroid after making the initial assignments and then iteratively refines the assignments. After then, the centroid is relocated to the new mean location, and the data points are assigned to the closest. Until the centroids stop moving altogether or a certain amount of iterations have been completed, this process is repeated. The algorithm has now reached its convergence point, and the resulting clusters show collections of data points that are related to one another in terms of their separation from the centroids.

Algorithm for K-means Clustering :

- Calculating K, the total amount of clusters.
- Choose K data points at random and assign each one to a cluster.
- Continue working through the previous steps to achieve the ideal centroid, which is the centroid's data point that stays the same. .
- The squared distances between the centroids and the data points are added up and calculated.
- The cluster that contains each data point should be chosen based on its proximity to the other clusters. Calculate the centroids for clusters by averaging all of the data points within the cluster.

Pseudo code of the K-means Clustering algorithm :

- Reading the dataset containing the word embeddings is the initial step. Using the "with open" statement and the read lines method, this is accomplished. After that, the dataset is kept in the array.
- The function compute silhouette coefficient is described. This function figures out the silhouette coefficient for a set of labels and data points. In a clustering technique, the silhouette coefficient

measures how well-separated the clusters are.
- The k-means clustering algorithm is implemented in this function. The input consists of the data, the number of clusters to be formed, and the maximum number of algorithm iterations to be performed.
- Initializing k centroids at random using the data is the first step in the k-means algorithm.
- The method then updates the centroids with the mean of the data points assigned to them after iteratively assigning each data point to its nearest centroid.
- The silhouette coefficient for the final clustering is generated once the method has run for the maximum number of iterations or until convergence is reached.
- The code then uses the matplotlib plotting library to display the silhouette coefficient for various k values. The generated plot can be used to select the ideal number of clusters from the dataset.

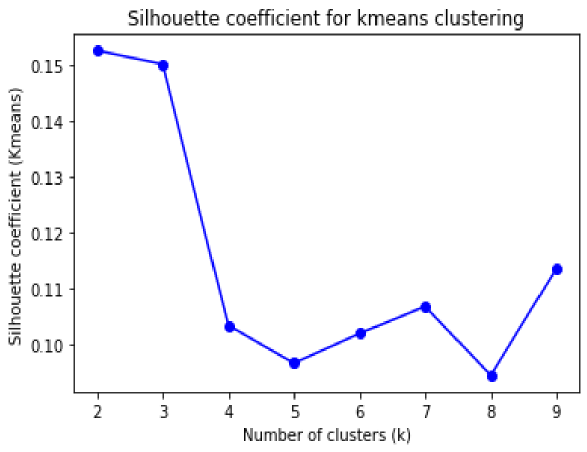
k-means++ clustering algorithm

The k-means++ algorithm is a development of the k-means method that seeks to enhance centroids initialization. The k-means algorithm's initialization step involves selecting k locations at random from the dataset to serve as the initial centroids. The initial centroids are selected more intelligently using the k-means++ method. To be more precise, the algorithm begins by randomly choosing one point from the dataset to act as the first centroid. The method next samples a point from the dataset with a probability proportional to the square of its distance from the nearest existing centroid for each consecutive centroid. With a wider initial distribution of centroids and hence better clustering outcomes, this method enhances the likelihood of choosing sites that are distant from the current centroids.

Algorithm for K-means++ Clustering :
- At first, decide by random which data point's centroid to use.
- Calculate the distance between each data point and the nearest centroid.
- Select the next centroid from the data points with a probability proportional to the square of how far away the previous centroid is from the current centroid.
- Continue with steps 2 and 3 for selecting k centroids.
- The data can be grouped using the k-means method once the centroids have been seeded with k-means++.

Pseudo Code of the K-means++ Clustering algorithm :
- This stage involves reading the word embedding dataset, which is essentially a numerical representation of a word used in natural language processing tasks
- Defining the algorithm and its related functions. The centroids are initialized, updated, and new data point labels are predicted using these routines, which also carry out the basic logic of the K-means++ method.
- In order to initialise the centroids in a way that increases the algorithm's effectiveness and efficiency, this stage entails applying the technique. In particular, we use the K-means++ initialization method to choose the initial centroids by considering the separations between the data points and the current centroids.
- The algorithm implemented in this stage, which entails updating the centroids iteratively and reassigning data points to the closest centroid until convergence or a predetermined number of iterations is attained.
- The silhouette score calculation, which is a measure of how similar a data point is to its own cluster in comparison to other clusters, is put into practise in this stage.
- The loaded dataset is subjected to the in the final stage, and its performance is assessed using the silhouette score. This score can be used to establish the ideal number of clusters for the dataset and then visualise the resulting clusters to reveal hidden data patterns.

Bisecting k-means clustering algorithm



Bisecting k-Means is similar to the k-Means algorithm in that it creates clusters of data using a hierarchy of centroids rather than a single set of centroids. This algorithm's fundamental concept is to start with a single cluster that contains all the data points and recursively split the biggest cluster until k clusters are generated. The Bisecting k-Means algorithm is superior to the traditional k-Means algorithm in a number of ways. It creates a hierarchical structure of the clusters, which is helpful for figuring out how the clusters are related to one another. It can handle non-convex clusters since it employs a hierarchical strategy to cluster the data, and because it starts with a single cluster including all the data points, it is less sensitive to the initial selection of centroids. The necessity to repeat the k-Means method numerous times in order to create k clusters makes the approach computationally more expensive than the normal k-Means algorithm.

Algorithm for Bisecting K-means Clustering :

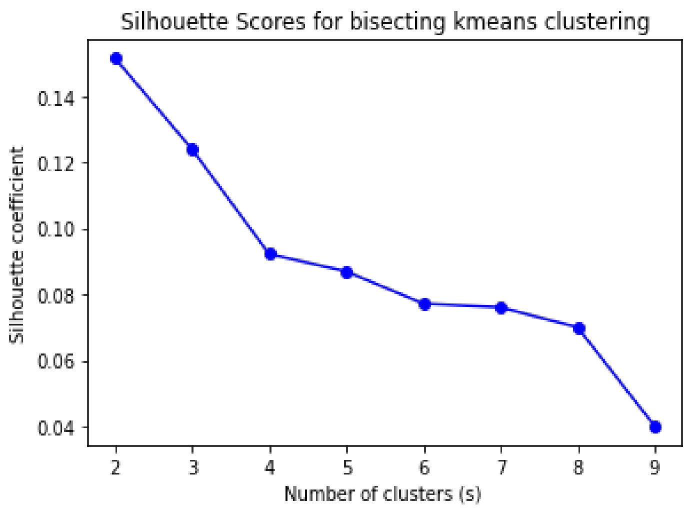
- Collect all the data points into one cluster.
- Continue working up until the required number of clusters is reached.
- Apply the k-means clustering algorithm with k=2 to the selected cluster
- Determine the SSE (sum of squared errors) for each cluster that is generated.
- Decide which cluster has the highest SSE, then replace it with the most current clusters.

Pseudo Code of the Bisecting K-means Clustering algorithm :

- Initialization of arguments for the BisectingKMeans class specify the desired number of clusters and the maximum number of iterations for the k-Means method, respectively.
- The list is initialised by the given method with the dataset as its only constituent.
- The algorithm selects the cluster with the highest SSE at each iteration and divides it into two clusters using the k-Means technique.
- The kmeans function is used on the selected cluster to split the cluster. This function randomly chooses two centroids as initial values and then clusters the cluster using k-Means till convergence. Two new clusters are formed using the resulting centroids.
- The two new clusters in the list then take the place of the initial cluster, and the loop is repeated until the required number of clusters is reached.
- In addition to calculating the SSE for the obtained centroids, the kmeans function also returns the centroids together with the SSE value.
- The list containing the centroids of the last clusters is what the algorithm finally produces.

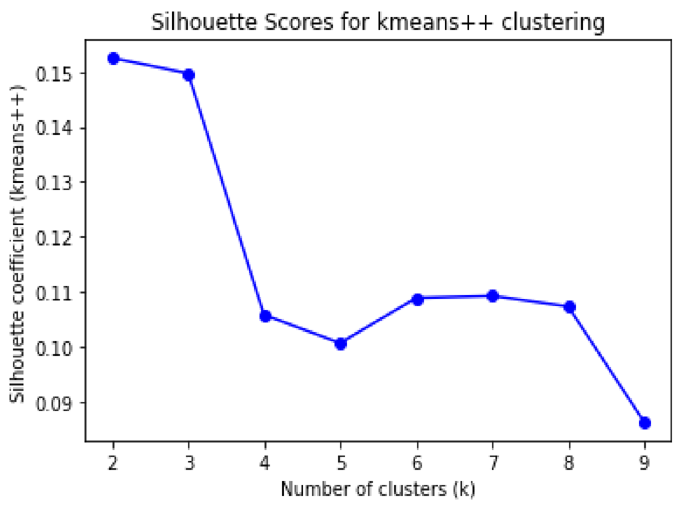
The first part's Kmeans clustering technique was successfully applied using the provided dataset. The code skips the value 1 and starts from 2 because K can range from 1 to 9, with 1 being the initial cluster with just 1 cluster. With the number of clusters in the X axis and the silhouette coefficient in the Y axis, the silhouette coefficient for each set of clusters is plotted.

- Silhouette Coefficient = (b - a) / max (a, b)



According to the provided dataset, the K-Means++ clustering algorithm described in part 2 is effectively implemented. K can be a value between 1 and 9, with 1 being the initial cluster with only 1 cluster. The codeskips value 1 and begins at value 2. Each set of clusters' silhouette coefficient is plotted, with the number of clusters in the X axis and the silhouette coefficient in the Y axis.

- Silhouette Coefficient = (b - a) / max (a, b)



A hierarchy of clustering is successfully computed using the Bisecting k-Means technique that was created in part 3. Using this, the initial single cluster is refined into nine clusters. The hierarchical structure of the clusters of data points is obtained using the bisecting k-means technique. In comparison to earlier K-Means implementation, this provides more information. As stated in the question, the s-clusters and the silhouette coefficient are plotted on the same graph with the X and Y axes, respectively.

According to our dataset, as we know the number of clusters should be 1 to 9, considering the differentclusterings that is obtained, the best clustering for the given dataset is as below

kmeans = 2	Silhouette coefficient = 0.15252
kmeans = 3	Silhouette coefficient = 0.15006
kmeans = 4	Silhouette coefficient = 0.10330
kmeans = 5	Silhouette coefficient = 0.09657
kmeans = 6	Silhouette coefficient = 0.10186
kmeans = 7	Silhouette coefficient = 0.10671
kmeans = 8	Silhouette coefficient = 0.09437
kmeans = 9	Silhouette coefficient = 0.11350

kmeans++ = 2	Silhouette coefficient = 0.15252
kmeans++ = 3	Silhouette coefficient = 0.14973
kmeans++ = 4	Silhouette coefficient = 0.10576
kmeans++ = 5	Silhouette coefficient = 0.10061
kmeans++ = 6	Silhouette coefficient = 0.10883
kmeans++ = 7	Silhouette coefficient = 0.10922
kmeans++ = 8	Silhouette coefficient = 0.10729
kmeans++ = 9	Silhouette coefficient = 0.08624

bisectingkmeans = 2	Silhouette coefficient = 0.15151
bisectingkmeans = 3	Silhouette coefficient = 0.12393
bisectingkmeans = 4	Silhouette coefficient = 0.09222
bisectingkmeans = 5	Silhouette coefficient = 0.08695
bisectingkmeans = 6	Silhouette coefficient = 0.07720
bisectingkmeans = 7	Silhouette coefficient = 0.07607
bisectingkmeans = 8	Silhouette coefficient = 0.07003
bisectingkmeans = 9	Silhouette coefficient = 0.04010

When comparing the three methods' Silhouette coefficients, we can see that k-means and k-means++ constantly exceeds Bisecting k-Means. This suggests that Bisecting k-Means perform poorly than k-means and k-means++ at clustering this dataset. We can see that for all values of k-means and k-means++ have the same Silhouette coefficients. This demonstrates that in this dataset, random initialization did not significantly improve clustering accuracy when centroids were initiated using k-means++.

kmeans clustering algorithm - Silhouette coefficient : 0.10431926628978942
kmeans++ clustering algorithm - Silhouette coefficient : 0.08624371476128759
Bisecting kmeans algorithm - Silhouette coefficient : 0.04010868121607036

We may conclude that **k-means** is the most effective method for grouping this dataset based on the Silhouette coefficients