In [40]:

```python
# Tweets KMeans Clustering by:
# Siddhant Medar (SSM200002)
# Adithya Iyer (ASI200000)

import math
import random
import re
import copy
import urllib
import urllib.request


# Data Preprocessing
def preprocess(p):
    twt = []
    f = urllib.request.urlopen(p)

    lines = [x.decode("cp1252").strip() for x in f]
    for ln in lines:
        # tw = ln.split('|')[2].split('http://')[0]
        # tw = re.sub('@|#', '', tw)
        tw = ln.split('|')[2]
        tw = re.sub('via @\S+', '', tw)
        tw = re.sub('@\S+', '', tw)
        tw = re.sub('#', '', tw)
        tw = re.sub('http:\/\/\S+', '', tw)
        tw = tw.lower()
        twt.append(tw)

    return twt


# Calculating Jaccard Distance
def distance(x, y):
    intersection = list(set(x) & set(y))
    intrs = len(intersection)
    """Union = list(set(x) | set(y))
    un_ln = len(Union)"""
    un_ln = len(x)+len(y) - intrs
    overlap = (float(intrs) / un_ln)
    return round(1 - overlap, 4)


# output in the form of cluster no. and the number of tweets in that cluster.
def clusters_output(cluster_no, k, twt_id):
    cluster_id = []
    print("Cluster Number : Number of Tweets")
    for i in range(k):
        cluster_id.append([j for j, c in enumerate(cluster_no) if c == i])
        tw_no = [y for y in cluster_id[i]]
        print(i + 1, ":", len([twt_id[y] for y in tw_no]))


# computing the sum of squared errors
def SSE(centroids, tweet_text, k):
    sum_sq_error = 0
    for i in range(k):
        for j in range(len(tweet_text)):
            dist_from_i = distance(tweet_text[j], centroids[i])
            dist_from_x = dist_from_i
            for x in range(k):
                dist_from_x = min(dist_from_x, distance(tweet_text[j], centroids[x]))
            if (dist_from_i==dist_from_x):
                sum_sq_error += math.pow(dist_from_i, 2)
    print(sum_sq_error)


# updating the centroids at every iteration
def update_centroid(twt_id, cluster, tweet_text, k):
    ind = []
    upd_centroid_tweet_id = []

    for i in range(k):
        ind.append([j for j, cls in enumerate(cluster) if cls == i])
        cl = ind[i]

        # cl gives the indices of the elements of every cluster k

        if len(cl) != 0:
            tweet = [tweet_text[s] for s in cl]
            similarity_distance = [[distance(tweet[i], tweet[j]) for j in range(len(cl
))] for i in range(len(cl))]

            total_similarity = [sum(i) for i in similarity_distance]
            # idx of the point closer to all the other points
            upd_centroid_tweet_id.append(cl[(total_similarity.index(min(total_similarit
y)))])
    updated_twt_centroid = [twt_id[x] for x in upd_centroid_tweet_id]
    return updated_twt_centroid


# k-means clustering implementation from scratch
def kmeans(twt_id, centroids, twt_txt, ln, k):
    count = 0
    for itr in range(50):
        count = count + 1
        cluster_id = []
        cnt = 0
        for i in range(ln):
            cnt += 1
            dist = [distance(twt_txt[i].split(' '), centroids[j].split(' ')) for j in r
ange(k)]
            dist_id = dist.index(min(dist))
            cluster_id.append(dist_id)

        nw_centroid = update_centroid(twt_id, cluster_id, twt_txt, k)
        print("Updated Centroids After Iteration ", itr + 1)
        print(nw_centroid)
        cen = [twt_txt.index(item) for item in centroids]
        if cen == nw_centroid:
            print("KMeans Clustering converged after iteration no. = {} for K = {}".for
mat(itr + 1, k))
            break
        centroids_id = copy.deepcopy(nw_centroid)
        centroids = []
        for twt_idx in centroids_id:
            centroids.append(twt_txt[twt_idx])

        clusters_output(cluster_id, k, twt_id)
        print("Sum of squared Error (SSE) after iteration no.", itr + 1)
        SSE(centroids, twt_txt, k)


if __name__ == '__main__':
    path = "https://raw.githubusercontent.com/siddhantmedar/CS6375-Machine-Learning/mai
n/Tweets/foxnewshealth.txt"
    # path = "https://raw.githubusercontent.com/siddhantmedar/CS6375-Machine-Learning/m
ain/Tweets/msnhealthnews.txt"
    # path = "https://github.com/siddhantmedar/CS6375-Machine-Learning/blob/main/Tweet
s/usnewshealth.txt"

    # k values tried = [1,2,4,5,8,9,10]
    k = 4
    # preprocess the tweets
    tweets = preprocess(path)
    length = len(tweets)
    twt_ids = []
    for i in range(len(tweets)):
        twt_ids.append(i)
    tweets_centroid = random.sample(tweets, k)

    # run k-means clustering on the tweets
    kmeans(twt_ids, tweets_centroid, tweets, length, k)
```

```
Updated Centroids After Iteration  1
[1, 247, 277, 58]
Cluster Number : Number of Tweets
1 : 470
2 : 1085
3 : 200
4 : 245
Sum of squared Error (SSE) after iteration no. 1
1245.6720947600006
Updated Centroids After Iteration  2
[916, 1755, 54, 58]
Cluster Number : Number of Tweets
1 : 1649
2 : 26
3 : 258
4 : 67
Sum of squared Error (SSE) after iteration no. 2
1250.8223036000002
Updated Centroids After Iteration  3
[916, 247, 54, 58]
Cluster Number : Number of Tweets
1 : 249
2 : 1515
3 : 148
4 : 88
Sum of squared Error (SSE) after iteration no. 3
1243.0455140200004
Updated Centroids After Iteration  4
[916, 247, 54, 58]
KMeans Clustering converged after iteration no. = 4 for K = 4
```