Improved K-means in NLP Application

——— **Finding clusters of related words**

Abstract

In my final technical project, An improved K-means methods is applied to cluster the related words in books' title. In this report, the substantial technique improvement would be introduced and several experiment will show the new algorithms' power. In the end, I combine the traditional Natural language Processing methods and the new algorithm to build a better tool on words classification.

1. Introduction: Motivation.

Unstructured data in the form of text is everywhere: emails, chats, web pages, social media, support tickets, survey responses, and more. Text can be an extremely rich source of information, but extracting insights from it can be hard and time-consuming due to its unstructured nature. Businesses are turning to text classification for structuring text in a fast and cost-efficient way to enhance decision-making and automate processes.

Sometimes, we classify the words and text using supervised learning to label the words, but labeling is pretty expensive and time-consuming. Therefore, clustering technique is introduced to do the work after feature extraction.

However, traditionally, if we use original K-means clustering to classify the word, we cannot get the accurate and precise results because the original k-means cannot deal with problems like the elliptical problem, which means we need to improved the clustering methods in the final words classification stage.

2. Problem description: How does the traditional K-means fail?

If we want to apply the original K-means, we surly cannot obtain the satisfactory classification result brought by its drawbacks. We can see disadvantages below:

- **Being dependent on initial values.**

For a low k, we can mitigate this dependence by running k-means several times with different initial values and picking the best result. But for a engineering project, it is pretty time-consuming.

- **Trouble in Clustering data of varying sizes and density.**

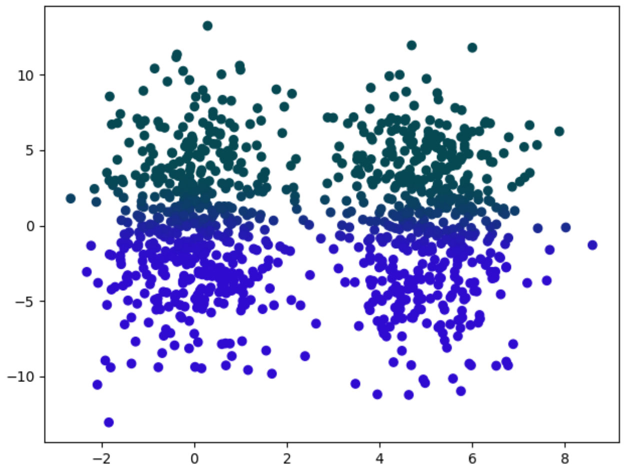


Fig 1: failed case 1

k-means has trouble clustering data where clusters are of varying sizes and density. Since it tries to minimize the within-cluster variation, it gives more weight to bigger clusters than smaller ones. In other words, data points in smaller clusters may be left away from the centroid in order to focus more on the larger cluster.

- **Misclustering in the uniform distribution.**

K-means may still cluster the data even if it can't be clustered such as data that comes from uniform distributions.

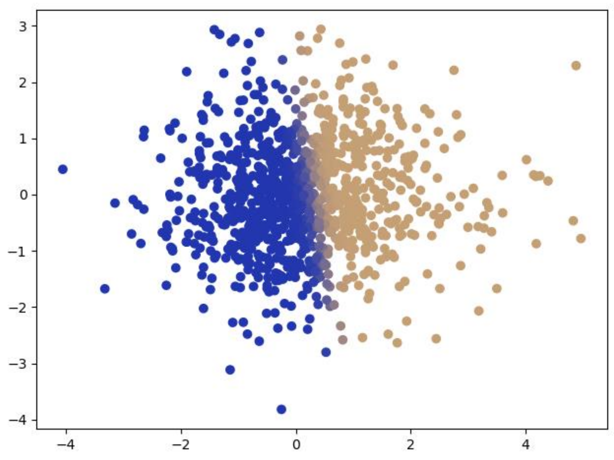
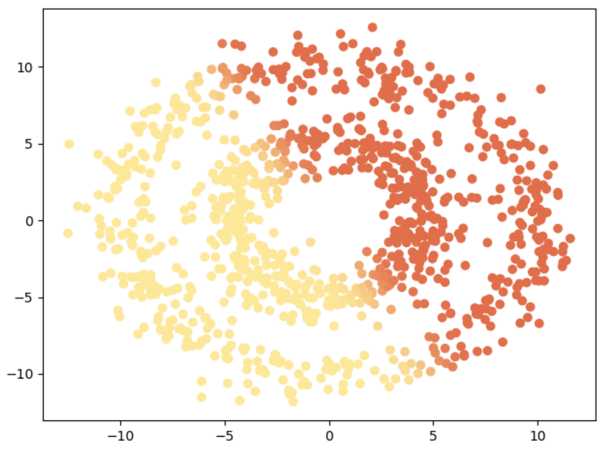


Fig 2&3: failed case 2&3



- **Cannot deal with the complicated geometric shapes**

- **different measurement units of data features and different distance functions adopted will affect the clustering results.**

  Hence, feature normalization is required;

- **generally, the influences of dimensional features are different in computing the distances of one datum and the cluster centers**.

Hence, applying appropriate weights for the data features is necessary; The feature we talked about below would influence the results negatively, and I will show the result in the chapter 5.

3. Baseline method

If we choose unsupervised learning in the labeling steps, our architecture is as below:

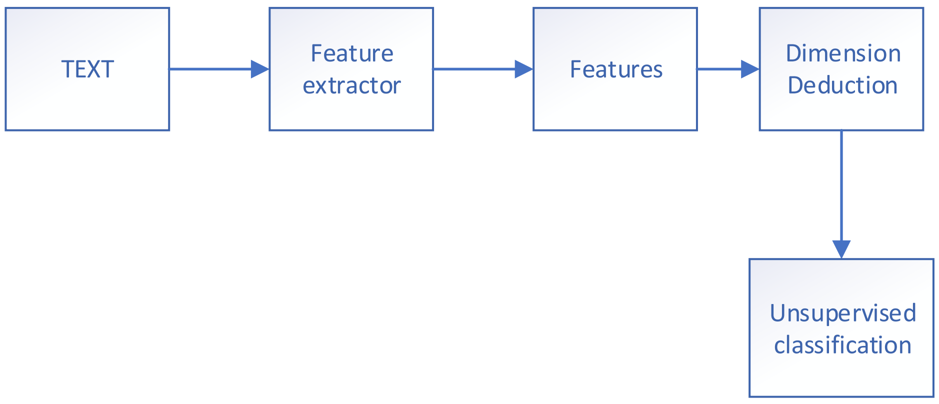


Fig 4: Architecture of words classification

1. Feature Extraction using TF-IDF:

TF-IDF is a very interesting way to convert the textual representation of information into a Vector Space Model (VSM), or into sparse features. It is based on the assumption that less frequently occurring words are more important.

Term frequency-Inverse document frequency uses all the tokens in the dataset as vocabulary. Frequency of occurrence of a token from vocabulary in each document consists of the term frequency and number of documents in which token occurs determines the Inverse document frequency. What this ensures is that, if a token occurs frequently in a document that token will have high TF but if that token occurs frequently in majority of documents then it reduces the IDF ,so stop words like "*an, the, I* ", which occur frequently are penalized and important words which contain the essence of document get a boost. Both these TF and IDF matrices for a particular document are multiplied and normalized to form TF-IDF of a document:

$$W_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

2. Dimension Reduction: t-SNE

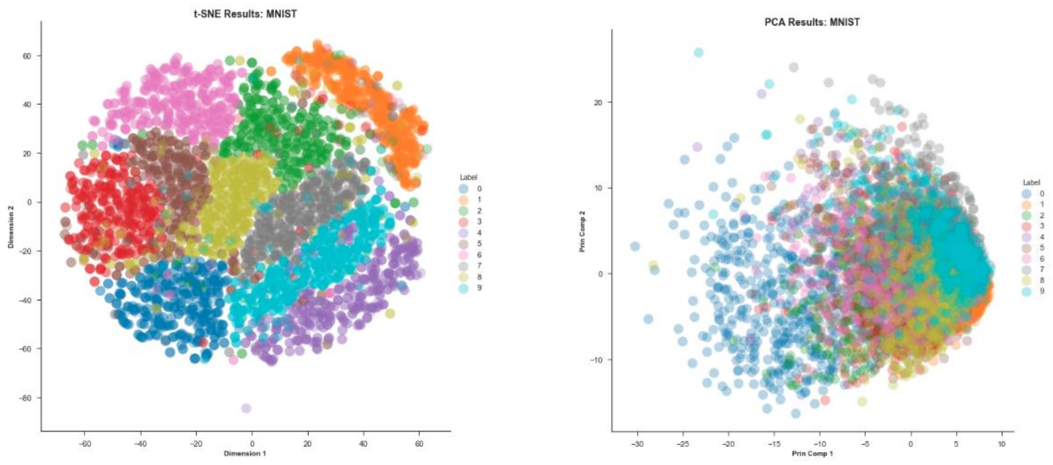


Fig 5: t-SNE and PCA on MNIST dataset

In order to visualize our classification, we need to deduce the data's dimension, here we apply the t-SNE. t-Distributed Stochastic Neighbor Embedding (t-SNE) is an unsupervised, non-linear technique primarily used for data exploration and visualizing high-dimensional data. In simpler terms, t-SNE gives you a feel or intuition of how the data is arranged in a high-dimensional space.t-SNE differs from PCA by preserving only small pairwise distances or local similarities whereas PCA is concerned with preserving large pairwise distances to maximize variance.

The algorithm starts by calculating the probability of similarity of points in high-dimensional space and calculating the probability of similarity of points in the corresponding low-dimensional space. The similarity of points is calculated as the conditional probability that a point A would choose point B as its neighbor if neighbors were picked in proportion to their probability density under a Gaussian (normal distribution) centered at A.

It then tries to minimize the difference between these conditional probabilities (or similarities) in higher-dimensional and lower-dimensional space for a perfect representation of data points in lower-dimensional space.

To measure the minimization of the sum of difference of conditional probability t-SNE minimizes the sum of Kullback-Leibler divergence of overall data points using a gradient descent method.

3. Traditional k-means to cluster.

After we obtain the two dimensions data, we use traditional soft K-means to classify. The algorithm is as below:

Input: dataset S, and the desired number K of clusters.

Output: K disjointed clusters

(1) K data are randomly chosen from S as the cluster centers of the K clusters.

(2) Assort each datum to the cluster whose center it is the shortest distance from.

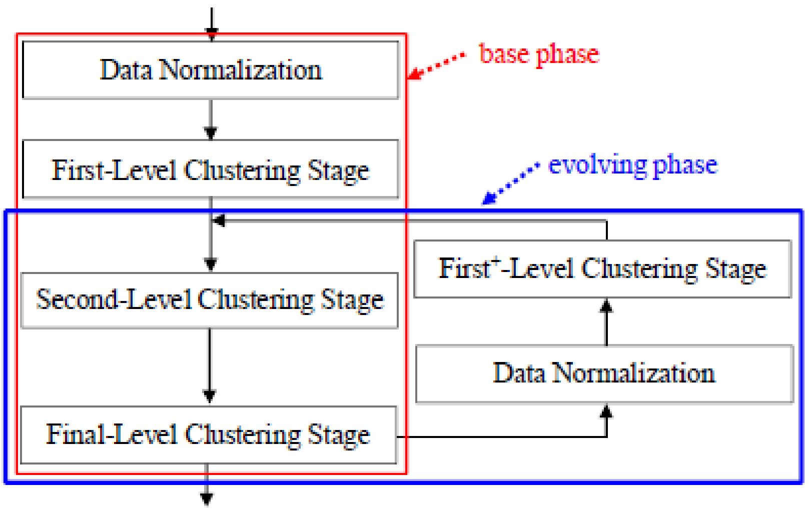
(3) Recalculate the center of each cluster only based on the data in the cluster.

(4) When the new cluster centers are the same as the cluster centers obtained in the previous iteration, output the clustering results; otherwise, iterate from Step (2).The k-means algorithm is simple and efficient.

4. New method: using improved Tri-level K-means algorithms.

Our new method shares the same feature extraction step and the dimension reduction step as original one. For the k-means clustering part, we change it into a better version named Tri-level K-means algorithm.

4.1 Tri-level K-means architecture:



Data Normalization
First-Level Clustering Stage
Second-Level Clustering Stage
Final-Level Clustering Stage
First⁺-Level Clustering Stage
Data Normalization
base phase
evolving phase

Fig 6: Architecture of Tri-level K-means

Tri-level k-means consists of: Data normalization, first level clustering second level clustering, and final-level clustering. First-level: divide the data into k′<k. Using the standard deviation of data to decide the number clustering should fully divided.

4.1.1 Data Normalization:

Normalization For the clustering assignment based on the distance between two data, the features with smaller variations are more influential than the features with bigger variations. Normalizing the feature values of each dimension is hence necessary.

Consider the dataset $S' = \{X_1', X_2' \ldots X_N'\}$ comprised of N data, where X_i' is the i_{th} datum in S' and $(X_{i1}', X_{i2}' \ldots X_{id}')$ are the features of X_i'. Each feature value X_{ij}' will be normalized into X_{ij} by the following equation:

$$X_{ij} = \frac{x_{ij}' - MIN(x_{kj}')}{MAX(x_{kj}') - MIN(x_{kj}')}$$

Let X_i, consisting of $(X'_{i1}, X'_{i2} \dots X'_{id})$, be the normalized datum of X'_i, Hence, $S' = \{X'_1, X'_2 \dots X'_N\}$ is changed in to $S = \{X_1, X_2 \dots X_N\}$

4.1.2 First-Level clustering:

The basic principle of data clustering is that the data with huge discrepancy should be assigned to different clusters and a cluster with an enormous number of data needs to be divided into smaller ones. Standard deviation is frequently used to measure the discrepancy of data in a cluster. Hence, for a big cluster with a larger quantity and a larger standard deviation of data, the tri-level k-means algorithm will divide the big cluster into more small clusters, and vice versa. Let $(X_{ci1}, X_{ci2} \dots, X_{cin})$ be the i_{th} datum in the c_{th} big cluster obtained in the first-level clustering stage. The goal of the second-level clustering stage is to partition the K' big clusters into K clusters according to the quantity and the standard deviation of the data in each big cluster where $K = k_1 + k_2 + k_3 + \cdots + k_k$.Let n_c be the number of data in the c_{th} big cluster, and Std_c be

$$Std_c = \frac{\sum_{j=1}^{d} \sqrt{\frac{\sum_{i=1}^{n_c} (x_{cij} - u_{cj})^2}{n_c}}}{d} , where\ u_{cj} = \frac{\sum_{i=1}^{n_c} x_{cij}}{n_c}$$

Then, the tri-level k-means algorithm classifies the data in each big cluster c into $K_c$clusters by a traditional k-means algorithm, where K_c is defined as follows:

$$K_c = \frac{n_c \times Std_c^e}{\sum_{i=1}^{k'} (n_i \times Std_i^e)} \times K$$

4.1.3 Final-Level clustering

In the second-level clustering stage, the tri-level k-means algorithm classifies the data in S into K clusters $(C_1, C_2 \dots C_K)$. Assume that$(X_{ci1}, X_{ci2,} \dots, X_{cik})$ are the features of the i_{th} datum in C_c and there are n_c data in C_c. Let

$$C_{cj} = \frac{1}{n'_c} \sum_{i=1}^{n'_c} x_{cij}$$

and $(C_{c1}, C_{c2}, C_{c3}, \dots, C_{cd})$ be the cluster center of cluster C_c. In this stage, the tri-level k-means algorithm assigns the cluster centers of $C_1, C_2, \dots C_k$ as the initial cluster centers and implements the traditional k-means algorithm to classify the data in S into K clusters. The traditional k-means algorithm is sensitive to initial cluster centers. For example, one intends to separate the data in Fig.7(a) into three clusters. Fig.7 demonstrates the processing of data clustering by traditional k-means algorithm, where the black dots are the data and the red dots represent the cluster centers. Fig. 7(d) shows the data divided into three clusters: two of them are close and include fewer data, while another cluster contains much more data. Fig. 8 shows the processing of clustering the same data in Fig. 7(a) by tri-level k-means algorithm. Fig. 8(a) are the data and the initial cluster centers. Fig. 8(b) shows the result obtained in the first-level cluster stage with $k' = \lceil \sqrt{k} \rceil = 2$ Compared to the result from Figs. 7(d), 8(e) displays that the clusters contain far closer quantities of data, and the data are much closer within the same cluster but much farther away from other clusters.

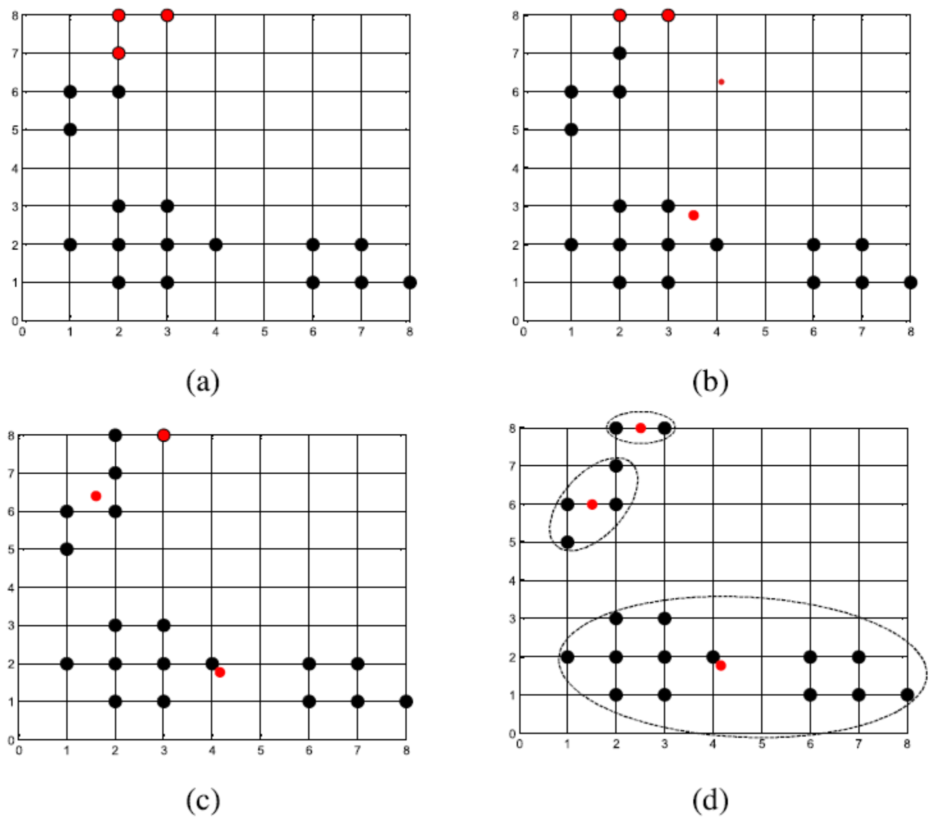


Fig 7: The processing of data clustering by traditional k-means algorithm

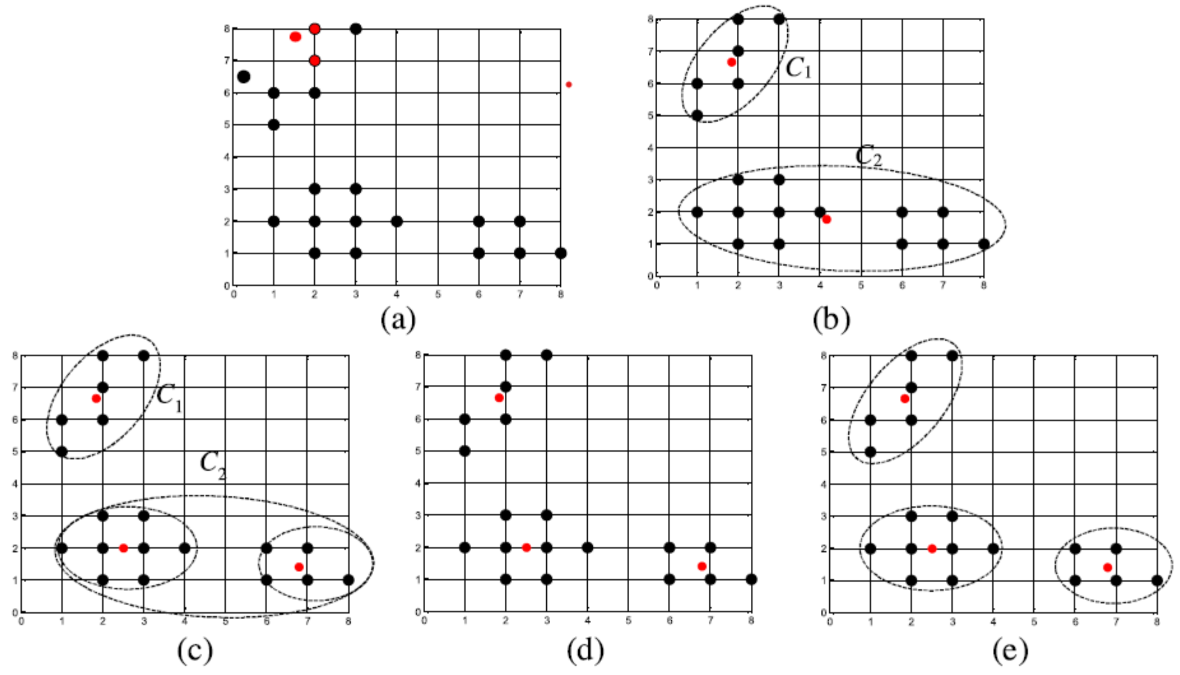


Fig 8: The processing of data clustering by Tri-level k-means algorithm

4.1.4 Time Complexity:

The tri-level k-means algorithm classifies the dataset S into K' big clusters, and then partitions each big cluster c into K_c clusters by traditional k-means algorithm. In order to facilitate the time complexity analysis, we assume that the data in S are uniformly assigned to each of K' big clusters in the first-level clustering stage. It means that each big cluster contains N/K' data ($n_1 = n_2 = \cdots = n'_k = N/K'$). It also supposes that the number of iterations the k-means algorithm executes is ITER. In each iteration the k-means algorithm runs, computing all the distances of each data in S and the cluster centers of the K' big clusters costs $N \times K' \times d$. Hence, the first-level clustering stage spends time $T_1 = N \times K' \times d$ ITER in total.

In the second-level clustering stage, the time to compute std_c and u_{cj} of each big cluster is 2 $\times$ $n_c \times$ d, where n_{cis} the number of data in the c_{th} cluster. Thus, this step takes computation time$\sum_{c=1}^{K'}(2 \times n_c \times d) = 2 \times N \times d$.

In the second-level stage, it spends time $d \times \frac{K}{K'}$ for computing the distances between a datum and all the cluster centers in a big cluster; there are $\frac{N}{K'}$ data in each big cluster, and there are K' big clusters, so the over-all time $T_2 = 2 \times N \times d + d \times \frac{K}{K'} \times \frac{N}{K'} \times ITER \times K' = d \times K \times \frac{N}{K'} \times ITER$ is utilized in this stage. Let $T = T_1 + T_2 = N \times d \times ITER \times \left(K' + \frac{k}{K'}\right).T$ is minimal if $k' = \sqrt{k}$. It means that when given $k' = \sqrt{k}$, the first-level clustering and the second-level clustering stages take the minimal time.

4.1.5 Online machine learning (OLML) based tri-level k-means algorithm

If the data in dataset S are frequently changed, the trained cluster centers cannot precisely describe the data in each cluster after a period of time; the cluster centers hence have to be updated. The tri-level k-means algorithm cannot efficiently deal with evolving data or frequently updated data. In this paper, the OLML based tri-level k-means algorithm hence is presented to solve this problem.

In the first-level clustering stage, the tri-level k-means algorithm classifies the data in S into K' big clusters. According to the quantity and the variation of the data in each big cluster, the number of clusters which each will be divided into is then derived. The tri-level k-means algorithm first partitions the data in S into K' big clusters by traditional k-means algorithm. The cluster center $(C_{c1}, C_{c2}, .. C_{cd})$ of each big cluster c is computed. The first-level clustering stage is a time consuming stage, where $T_1 = N \times K' \times d \times ITER$. The OLML based tri-level k-means algorithm will reduce the running time in evolving iterations.

The OLML based tri-level k-means algorithm contains two phases: the base phase and evolving phase. The base phase is similar to the tri-level k-means algorithm's included stages: data normalization, first-level clustering, second-level clustering, and final-level clustering. The evolving phase consists of the stages: data normalization, first+-level clustering, second-level clustering, and final-level clustering. The evolving phase re-clusters the data in dataset S when the content of S has been changed significantly. All the stages in the evolving phase are the same as those in the tri-level k-means algorithm except the first+-level clustering stage. The first+-level clustering stage is to renew the cluster centers of K' big clusters. Let $(C_{c1}, C_{c2}, \dots, C_{cd})$ be the old cluster center of big cluster c. First, it assorts each datum in S to the cluster with the shortest distance between the datum and each old cluster center. The j_{th} feature of the cluster center of big cluster c is then renewed into $C_{cj} = \frac{\sum_{i=1}^{n_c} x_{cij}}{n_c}$, where $(X_{i1}, X_{i2}, \dots, X_{id})$ is the i_{th} datum in the c_{th} big cluster. The time complexity of the first+-level clustering stage is $N \times K' \times d$, which is much less than $T_1 = N \times K' \times d \times ITER$.

4.2 The experiment result in Tri-level K-means algorithms.

Before we apply the improved algorithm, some experiments on small data and

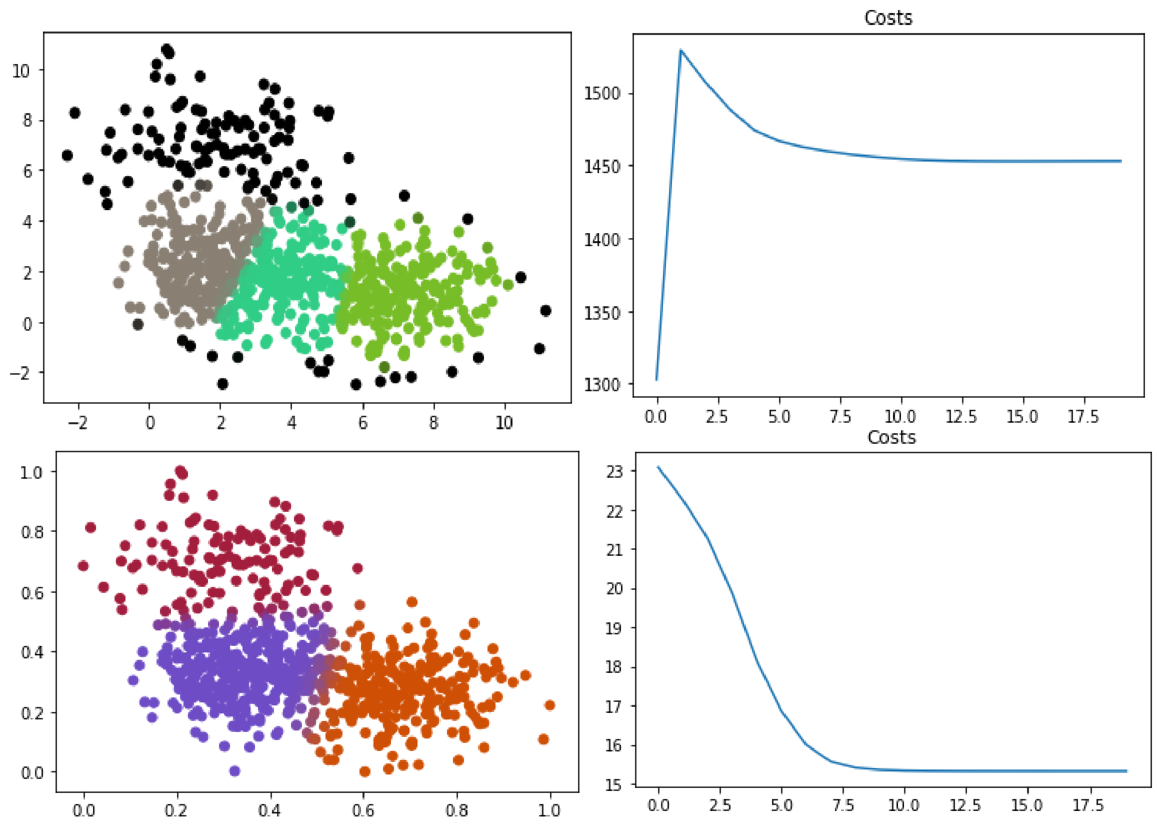


Fig9 result on the density-variant point

real data are necessary. Firstly, we use tiny different density points to do the experiment and we know that the Tri-level K-means can solve the problem related to density. Then, we use the MNIST dataset and Iris dataset

Table 1: Datasets feature

Dataset	**No.of samples**	**No.of Features**	**No.of Clusters**	**No.of Training samples**	**No.of Testing Samples**
Iris	150	4	3	75	75
Wine	178	13	3	90	88
Ecoli	336	7	8	170	166

Here is the results of K-means and Tri-level K-means on the different datasets, we can see the no matter which metric, Tri-level K-means performs significant improvements.

Table 2: The accuracies of classifying the testing datasets.

Dataset	**K-means**			**Tri-level K-means**		
	Precision	**Recall**	**F-measure**	**Precision**	**Recall**	**F-measure**
Iris	0.88	0.88	0.88	0.95	0.95	0.95
Wine	0.95	0.94	0.95	0.97	0.97	0.97
Ecoli	0.76	0.57	0.65	0.83	0.78	0.80

5. Experiment comparison on NLP Application

Finally, we bring our Tri-level K-means into the project and see all improvements in our application.

We use datasets from 2737 words from books titles. Firstly, we copy the tokenizer from sentiment example and add more stop words specific to words like 'introduction', 'edition', 'series', 'application', 'approach', 'card', 'access', 'package', 'plus'. Then we create a word-to-index map so that we can create our word-frequency vectors later. In addition, we also save the tokenized versions to create the input matrices.

Finally, we apply two algorithms and generate two pictures of clustering words.

From the pictures below, the Tri-level algorithms is far more regular and orderly through the observation. If we want to quantify, we could use a new metric named Davies-Bouldin Index. This is an internal evaluation scheme, where the validation of how well the clustering has been done is made using quantities and features inherent to the dataset. The score is defined as the average similarity measure of each cluster with its most similar cluster, where similarity is the ratio of within-cluster distances to between-cluster distances. Thus, clusters which are farther apart and less dispersed will result in a better score. The DBI for Tri-level K-means is 0.279 and the DBI for the K-means is 2.84. We can see the result of improved methods is 10 times better than the original one.

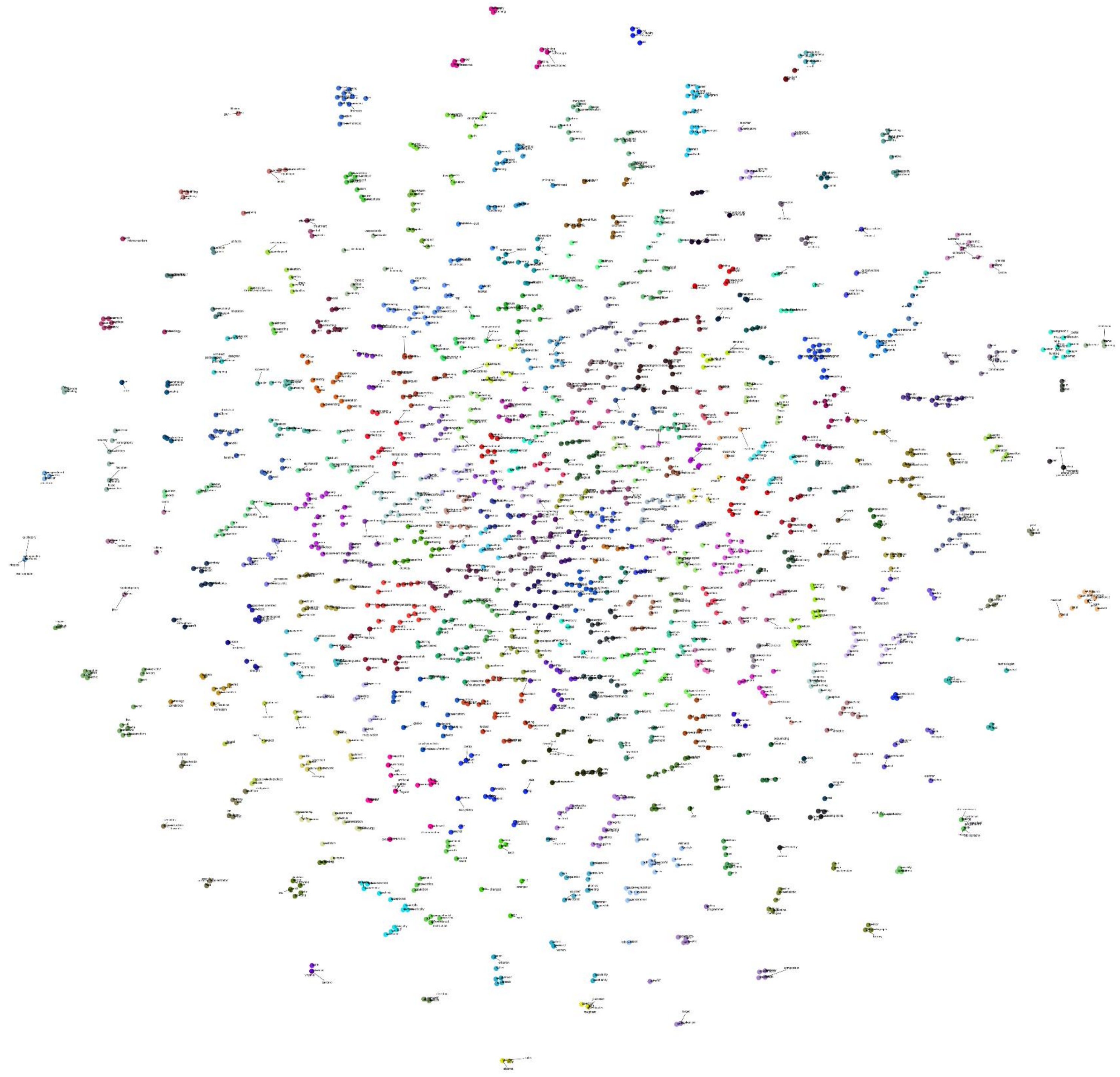


Fig10 Traditional K-means generated Words classification

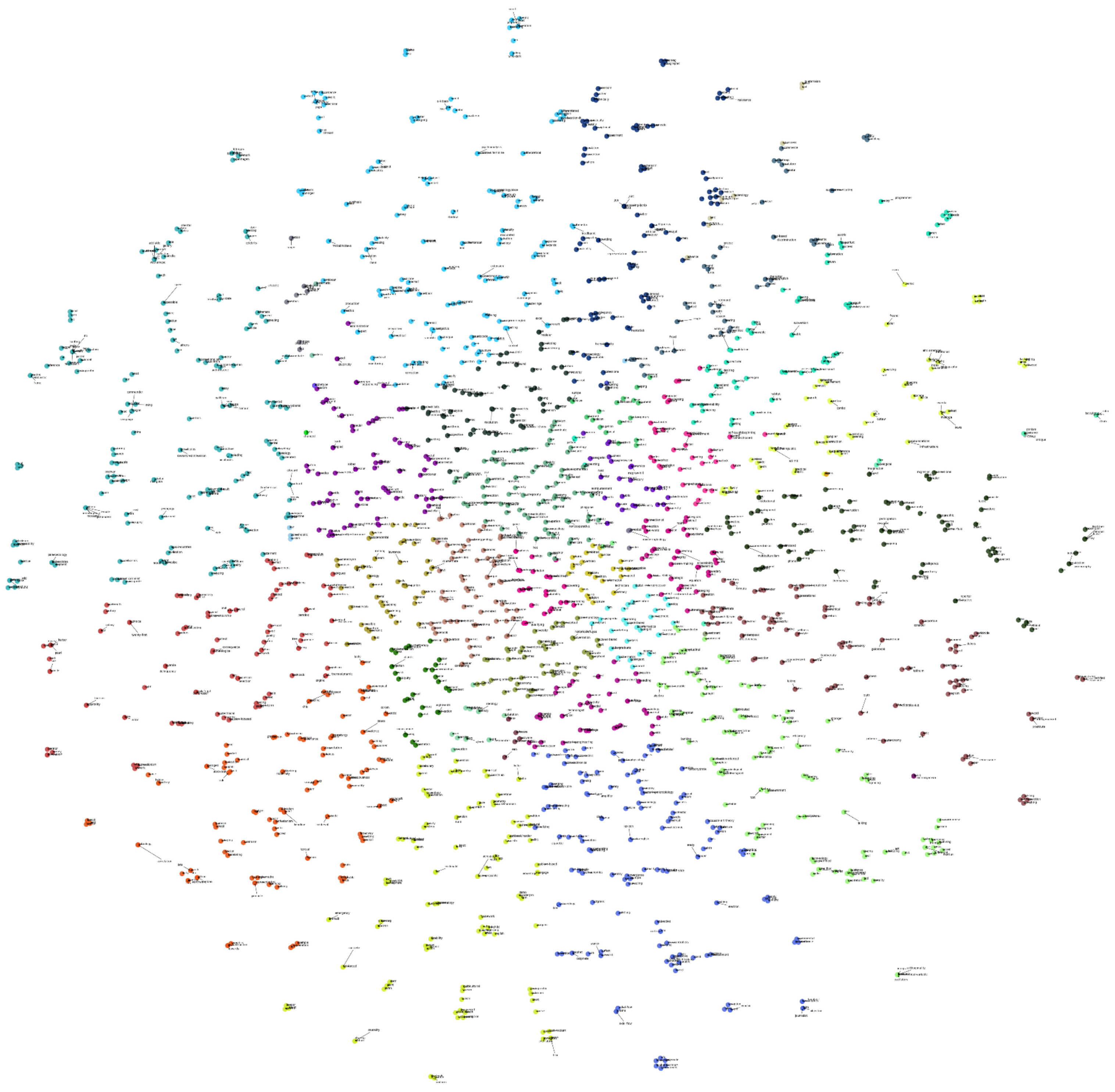


Fig11 Tri-level K-means generated Words classification

If we see the details of results, we would see two certain classification below:

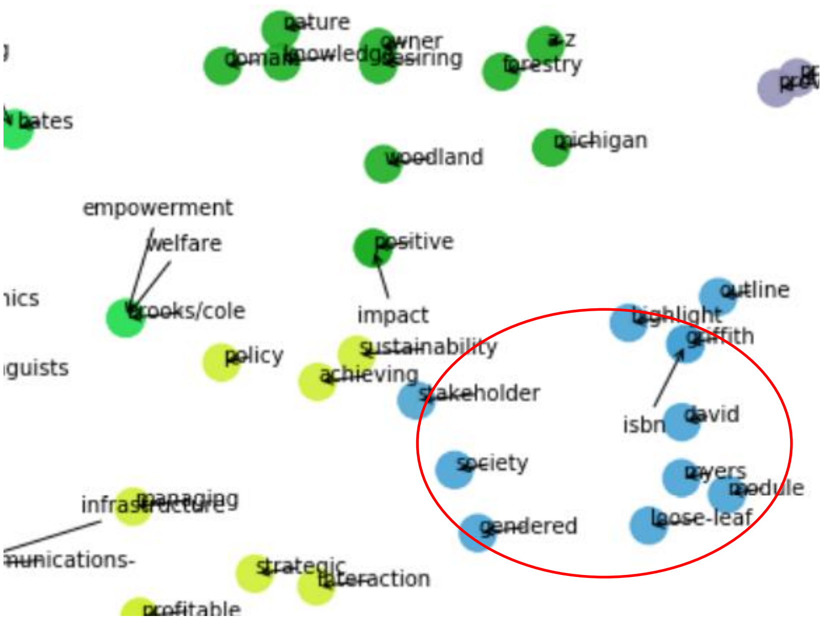


Fig11 Traditional K-means generated Words classification

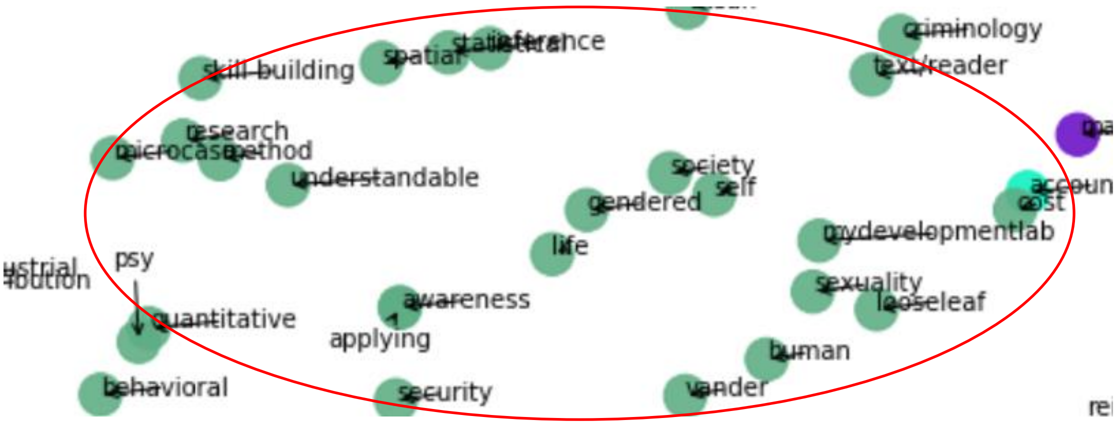


Fig12 Tri K-means generated Words classification

The same cluster notated by red circle has obvious difference. From human being’s perspective the second clouds’ words are more related and more accurate in the same class.

Using a more abundant and accurate context based on the better algorithms may be is the basis of creating more possibility in future.

6. Conclusion

My Projects is an improved version of words classification in NLP application. The basic work of words classification includes feature extraction using TF-IDF and dimension reduction like t-SNE. Based on this work, I implemented the Tri-level K-means to realized the final unsupervised classification.

From different metrics (F-measure, recall, precision, DBI), we can see the significant improvements and from the final visualized picture, we can obtain a direct view of how these two algorithms diverge and the advantages of Tri-level K-means algorithm.

7. Course feedback

This course named Statistical Pattern Recognition and it focus on a lot of topics in a wide range. I love this area, so I attend this course every time.

The instructor taught clearly and he is very humorous. The only thing I want to suggest is about the workload.

This course actually is totally ‘unsupervised learning’, so it all depends on the final work. I think it is better for student to write some view or report on the papers we should read so that students can be more hard-working.

In the end, I really want to thank professor Wu, because at first,I have no chance to register for limited quota. He gave me the chance to attend this course and I really learn a lot!

8. Referenced paper

[1] Shyr-Shen Yu, Shao-Wei Chu, Chuin-Mu Wang, Yung-Kuan Chan, ,Ting-Cheng Chang."

Two improved k-means algorithms". ELSEVIER Applied Soft Computing 68 (2018) 747–755