Indian Journal of Science and Technology, *Vol 7(6), 853–857, June 2014*
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Comparative Study of Fuzzy C Means and K Means Algorithm for Requirements Clustering

Ananthi Sheshasayee[1] and P. Sharmila[2*]

[1]Department of Computer Science and Applications, Quaid-e-Milleth Government College for Women, Chennai, Tamil Nadu, India.
[2]Department of Computer Applications, Vivekanandha Institute of Information and Management Studies, Tiruchengode, Tamil Nadu, India; dhyasharj@yahoo.co.in

Abstract

The Requirement Engineering is the most important phase of the software development life cycle which is used to translate the imprecise, incomplete needs and wishes of the potential users of software into complete, precise and formal specifications. These specifications can be decomposed on application of a data mining techniques, clustering. The process of clustering the requirements allows reducing the cost of software development and maintenance. In this research two most frequently used algorithms in clustering namely k means and fuzzy c means are used. The output generated is then analyzed for evaluating the performance of the two clustering algorithms. The requirements specified by the different stakeholders of the library are used as the input. The data mining tool WEKA was used for clustering. The clustering algorithms were then analyzed for accuracy and performance. On analysis the fuzzy c means algorithm was found to be more suitable for clustering of library requirements. The results proved to be satisfactory.

Keywords: Clustering, Data Mining, Fuzzy C Means, K Means, Library, Requirements

1. Introduction

Requirement engineering activities include elicitation, analysis and validation, and documentation of the collected requirements. Those finally collected requirements are entered in the Software Requirement Specification (SRS). These tasks individually contribute to the overall quality of the software. The ability of the requirement engineers in eliciting the requirements mainly depends upon the stakeholder's participation. Web based elicitation techniques such as forums, wikis and online survey are used now to elicit the requirements from a large number of stakeholders[1,2].

During the requirement elicitation, significant effort is needed in discovering and understanding the requirements from large number of stakeholders. As the knowledge of the stakeholders vary to a different extent care should be taken in choosing the elicitation methods while collecting the requirements. Some focus on a web-based toolset that helps requirements engineers identify project stakeholders, elicit product requirements and stakeholders' preferences for these requirements by asking stakeholders to recommend other stakeholders, propose new requirements, and rate already submitted requirements[3]. Unfortunately, this huge effort did not translate. Careful evaluation of the requirements is necessary as redundant requirements may be entered into the SRS. So prediction of valid requirements is necessary. This task when related to very large projects, are in need of automated support. The problem is how to automatically and efficiently coordinate large numbers of stakeholders' requests, and to arrange the subsequent requirements into meaningful structures.

Collected requirements are later prioritized based on factors like cost value and weights assigned to stakeholders group. Relative cost and value of each requirement

*Author for correspondence

is taken into consideration while eliciting the requirements from the stakeholders and later the requirements are prioritized based on the cost value[4]. Weights are also assigned to all stakeholders groups, by which we can compute the overall value of a requirement as the weighted sum of its value for each stakeholders group, and rank the set of requirements accordingly. Different variants of this approach are used in practice[5,6].

Clustering analysis as a data mining technique plays an important role in retrieving new pattern information. The information that is grouped in to clusters can later be further analyzed for any particular information. According to Guedalia et al.[7] cluster analysis serves as the pre-processing techniques for all other data mining operations. Clustering groups data objects which are similar in certain characteristic. Upon clustering, each group will contain objects which are similar[8,9]. Homogeneous stakeholders[10] are identified and grouped together on applying clustering techniques. These grouped stakeholders are then input for requirements selection and prioritization. The choice of applying particular clustering technique depends on various factors such as the desired output, type of data and the size of the data set, available hardware and software[13].

2. Proposed WORK

The proposed work decomposes the library requirements based on the common characteristic shared by the requirements using clustering technique. Thus the requirements that are grouped in each clusters exhibit certain properties that can be used for further software modernization, requirements re-use and software requirement improvement. Fuzzy c means and k means algorithm are the techniques applied for requirement clustering n this research work. The work has been carried out in WEKA. The Waikato Environment for Knowledge Analysis (WEKA) 3.60 serves as an intelligent tool for data analysis and predictive modelling. WEKA was chosen for its wide collection of free analytical tools and data mining algorithms (Figure 1).

3. Materials and Methods

Requirements clustering techniques address the relationship between requirements. Requirements clusters contribute to requirements reuse.

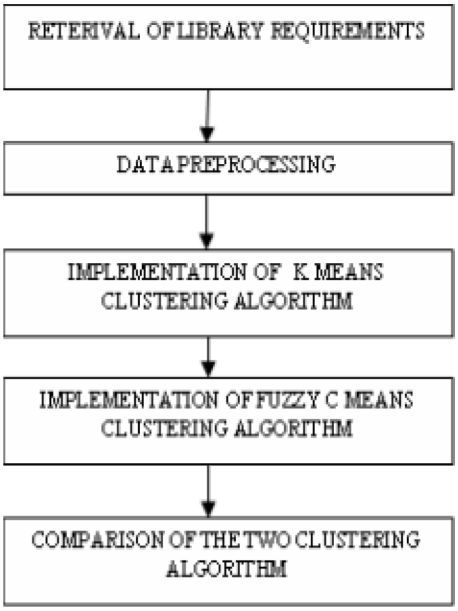

Figure 1. Proposed methodology for comparative analysis of clustering algorithm.

3.1 Data Preparation

The data set is taken from the UCI Machine Learning Repository which has a collection of number of datasets that is mostly used by the researchers of machine learning[14]. The requirements specified by the stakeholders of the library are taken as the data set.

3.2 Fuzzy C Means Algorithm

Fuzzy C-Means (FCM) is a method of clustering which allows one piece of data to belong to two or more clusters. This method (developed by Dunn[11] and improved by Bezdek[12]) is frequently used in pattern recognition. It is based on minimization of the following objective function:

$$J_m = \sum_{i=1}^{N}\sum_{j=1}^{C} u_{ij}^{m} \| x_i - c_j \|^2, 1 \le m < \infty \qquad (1)$$

where m is any real number greater than 1, u_{ij} is the degree of membership of x_i in the cluster j, x_i is the ith of d-dimensional measured data, c_j is the d-dimension center of the cluster, and $||*||$ is any norm expressing the similarity between any measured data and the center. Fuzzy partitioning is carried out through an iterative optimization of the objective function shown above, with the update of membership u_{ij} and the cluster centers c_j by:

$$u_{ij} = \frac{1}{\sum_{k-1}^{C}\left(\frac{\| x_i - c_j \|}{\| x_i - c_k \|}\right)^{\frac{2}{m-1}}}, \quad c_j = \frac{\sum_{i-1}^{N} u_{ij}^m . x_i}{\sum_{i-1}^{N} u_{ij}^m} \tag{2}$$

This iteration will stop when, $\max_{ij}\{|u_{ij}^{(k+1)} - u_{ij}^{k}|\} < \varepsilon$ where ε is a termination criterion between 0 and 1, whereas k is the iteration steps. This procedure converges to a local minimum or a saddle point of J_m.

The algorithm is composed of the following steps:

1. Initialize $U=[u_{ij}]$ matrix, $U^{(0)}$
2. At k-step: calculate the centers vectors $C^{(k)}=[c_j]$ with $U^{(k)}$

$$c_j = \frac{\sum_{i-1}^{N} u_{ij}^m . x_i}{\sum_{i-1}^{N} u_{ij}^m} \tag{3}$$

3. Update $U^{(k)}$, $U^{(k+1)}$

$$u_{ij} = \frac{1}{\sum_{k-1}^{C}\left(\frac{\| x_i - c_j \|}{\| x_i - c_k \|}\right)^{\frac{2}{m-1}}} \tag{4}$$

4. If $\|U^{(k+1)} - U^{(k)}\| < \varepsilon$ then STOP; otherwise return to step 2.

3.3 K Means Algorithm

K-means (MacQueen, 1967) is one of the simplest unsupervised learning algorithms that solve the well known clustering problem. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters) fixed a priori. The main idea is to define k centroids, one for each cluster. These centroids should be placed in a cunning way because of different location causes different result. So, the better choice is to place them as much as possible far away from each other. The next step is to take each point belonging to a given data set and associate it to the nearest centroid. When no point is pending, the first step is completed and an early group age is done. At this point we need to re-calculate k new centroids as barycenters of the clusters resulting from the previous step. After we have these k new centroids, a new binding has to be done between the same data set points and the nearest new centroid. A loop has been generated. As a result of this loop we may notice that the k centroids change their location step by step until no more changes are done. In other words centroids do not move any more. Finally, this algorithm aims at minimizing an *objective function*, in this case a squared error function. The objective function:

$$J = \sum_{j=1}^{k}\sum_{i=1}^{n} \| x_i^{(j)} - c_j \|^2 \tag{5}$$

where $\|x_i^{(j)} - c_j\|^2$ is a chosen distance measure between a data point $x_i^{(j)}$ and the cluster centre c_j, is an indicator of the distance of the n data points from their respective cluster centres.

The algorithm is composed of the following steps:

1. Place K points into the space represented by the objects that are being clustered. These points represent initial group centroids.
2. Assign each object to the group that has the closest centroid.
3. When all objects have been assigned, recalculate the positions of the K centroids.
4. Repeat Steps 2 and 3 until the centroids no longer move. This produces a separation of the objects into groups from which the metric to be minimized can be calculated.

4. Results and Discussion

The number of clusters is given by the user during the execution of the program. The output generates four clusters: cluster1, cluster2, cluster3 and cluster4. The cluster quality is evaluated using the number of clusters, execution time, f measure which is the combination of precision and recall ideas of information retrieval.

Distribution of requirements data set among the clusters: The total number of clusters specified by the user is four. Table 1 shows the total number of requirements that are distributed when both fuzzy c means and k means algorithm are applied.

Table 1. Distribution of requirements in clusters

Clustering algorithm	Cluster1	Cluster2	Cluster3	Cluster4
Fuzzy c means	201	145	225	223
K means	79	50	291	362

The distribution shows that the data points are uniformly distributed using fuzzy c means algorithm where the data points is above 200 in all clusters except cluster4. The data points are non-uniformly distributed using k means algorithm where the cluster4 has the maximum data points of 362 and cluster2 has the minimum data points of 50. Since the fuzzy c means algorithm is crisp, the data points that groups in one cluster also appear in other clusters also.

Calculation of execution time for fuzzy c means and k means algorithm: Table 2 shows the execution time taken by fuzzy c means and k means algorithm with increase in the number of records. The time consumption of fuzzy c means is less compared to the K-Means.

In Figure 2, the x-axis represents the number of records and the y-axis represents the time in milliseconds.

Calculation of precision, recall and f measure for fuzzy c means and K means algorithm: Table 3 shows the precision values, recall values and f measure values which evaluate the cluster accuracy using fuzzy c means and k means algorithm.

The clustering accuracy level identification using k means and Fuzzy C Means is shown in Figure 3. Fuzzy C Means technique increases the accuracy level compared to k-means.

Table 2. Execution Time Comparison (Milli Seconds)

Number of Records	Fuzzy-C-Means	K-Means
100	1614	1832
250	1623	1841
350	1648	1846
450	1656	1856
550	1668	1867
650	1689	1869
750	1678	1889

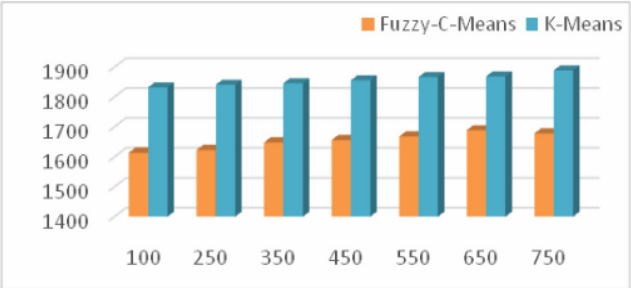

Figure 2. Time comparison of fuzzy c means and k means algorithm.

Table 3. Comparison of Precision, Recall and F values

Techniques Used	Precision	Recall	F-Measure
Fuzzy C Means	2.681592	0.72837836	1.1455898
K Means	0.4509804	0.3108108	0.36799997

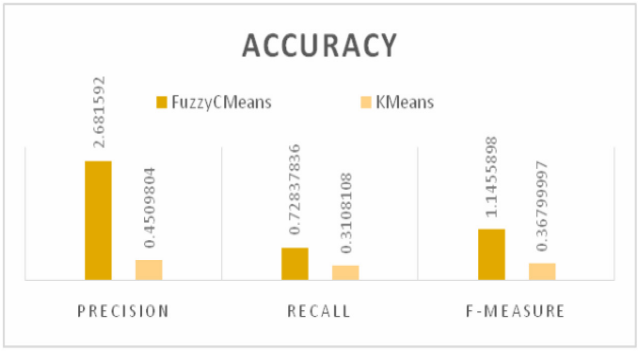

Figure 3. Comparison of precision, recall and f measure values of fuzzy c means and k means algorithm.

5. Conclusion and Future Scope

The proposed method groups the requirements using clustering method. After analysing both fuzzy c means and k means algorithm we conclude the following results. As the number of records increases the time execution of both the technique gets increased but the fuzzy c means performance is found to be better than the k means algorithm. The precision, recall and f measure values are more accurate on applying fuzzy c means compared to k means algorithm. The number of data points is evenly distributed in fuzzy c means algorithm. Fuzzy c means algorithm is efficient for larger data set and is well suited for requirement clustering. This work was intended in grouping the requirements where a large number of requirements are decomposed into small groups which can be easily analysed and further grouped. Future work may consider the stakeholders' priorities during requirement elicitation and the requirement clustering performance using various other clustering techniques can be compared.

6. References

1. Laurent P, Cleland-Huang J. Lessons learned from open source projects for facilitating online requirements processes. Requirements Engineering: Foundation for Software Quality. Springer Berlin Heidelberg; 2009. p. 240–55.

2. Lim SL, Quercia D, Finkelstein A. Stake Net: using social networks to analyse the stakeholders of large-scale software projects. Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering; 2010 May. 295–304.
3. Lim SL. Social networks and collaborative filtering for large-scale requirements elicitation [Doctoral Dissertation]. University of New South Wales; 2011.
4. Karlsson J, Ryan K. A cost-value approach for prioritizing requirements. Software. 1997; 14(5):67–74.
5. Robertson S, Robertson J. Mastering the requirements process. Harlow, UK: Addison Wesley; 1999.
6. Wiegers K. First things first: prioritizing requirements. Software Development. 1999; 7(9):48–53.
7. Guedalia ID, London M, Werman M. An on-line agglomerative clustering method for nonstationary data. Neural Computation. 1999; 11(2):521–40.
8. Barbará D, Chen P. Using self-similarity to cluster large data sets. Data Mining and Knowledge Discovery. 2003; 7(2):123–52.
9. Wei CP, Lee YH, Hsu CM. Empirical comparison of fast partitioning-based clustering algorithms for large data sets. Expert Systems with applications. 2003; 24(4):351–63.
10. Veerappa V, Letier E. Clustering stakeholders for requirements decision making. Requirements Engineering Foundation for Software Quality; 2011. p. 202–08.
11. Dunn JC. A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. 1973.
12. Bezdek JC. Pattern recognition with fuzzy objective function algorithms. Kluwer Academic Publishers; 1981.
13. Rao VS, Vidyavathi DS. Comparative Investigations and Performance Analysis of FCM and MFPCM Algorithms on Iris data. Indian Journal of Computer Science and Engineering. 2010; 1(2):145–51.
14. Asuncion A, Newman DJ. UCI Machine Learning Repository. University of California, School of Information and Computer Science, Irvine, CA; 2007.