STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume LI, Number 1, 2006

A COMPARISON OF CLUSTERING TECHNIQUES IN ASPECT MINING

GABRIELA ŞERBAN AND GRIGORETA SOFIA MOLDOVAN

Abstract. This paper aims at presenting and comparing three clustering algorithms in aspect mining: *k-means (KM)*, *fuzzy c-means (FCM)* and *hierarchical agglomerative clustering (HAC)*. Clustering is used in order to identify crosscutting concerns. We propose some quality measures in order to evaluate the results and we comparatively analyze the obtained results on two case studies.

Keywords: clustering, aspect mining.

1. Introduction

1.1. **Clustering.** Clustering is a division of data into groups of similar objects.

Clustering can be considered the most important *unsupervised learning* problem: so, as every other problem of this kind, it deals with finding a *structure* in a collection of unlabeled data.

Unsupervised classification, or clustering, aims to differentiate groups (classes or clusters) inside a given set of objects, with respect to a set of relevant characteristics or attributes of the analyzed objects. A *cluster* is, therefore, a collection of objects, which are similar between them and dissimilar to the objects belonging to other clusters.

Let $X = \{O_1, O_2, \ldots, O_n\}$ be the set of objects to be clustered. Using the vector-space model, each object is measured with respect to a set of l initial attributes $A_1, A_2, ..., A_l$ (a set of relevant characteristics of the analyzed objects) and is therefore described by a l-dimensional vector $O_i = (O_{i1}, \ldots, O_{il}), O_{ik} \in \Re, 1 \leq i \leq n, 1 \leq k \leq l$.

Received by the editors: May 24, 2006.

2000 *Mathematics Subject Classification.* 62H30, 68N99.

1998 *CR Categories and Descriptors.* I.5.3[**Computing Methodologies**]: Pattern Recognition – *Clustering*; D.2.7 [**Software Engineering**]: Distribution, Maintenance, and Enhancement –*Restructuring, reverse engineering, and reengineering.*

The measure used for discriminating objects can be any *metric* or *semimetric* function (d). In our approach we have used the *Euclidian distance*:

$$d(O_i, O_j) = d_E(O_i, O_j) = \sqrt{\sum_{k=1}^{l}(O_{ik} - O_{jk})^2}$$

The *similarity* between two objects O_i and O_j is defined as

$$sim(O_i, O_j) = \frac{1}{d(O_i, O_j)}$$

Many clustering techniques are available in the literature. Most clustering algorithms are based on two popular techniques known as *partitional* and *hierarchical* clustering ([4], [6] and [7]).

1.2. **Aspect Mining.** The Aspect Oriented Programming (AOP) is a new paradigm that is used to design and implement *crosscutting concerns* [9]. A *crosscutting concern* is a feature of a software system that is spread all over the system, and whose implementation is tangled with other features' implementation. Logging, persistence, and connection pooling are well-known examples of crosscutting concerns. In order to design and implement a crosscutting concern, AOP introduces a new modularization unit called *aspect*. At compile time, the aspect is woven to generate the final system, using a special tool called *weaver*. Some of the benefits that the use of AOP brings to software engineering are: better modularization, higher productivity, software systems that are easier to maintain and evolve.

Aspect mining is a relatively new research direction that tries to identify crosscutting concerns in already developed software systems, without using AOP. The goal is to identify them and then to refactor them to aspects, in order to achieve a system that can be easily understood, maintained and modified.

Crosscutting concerns in non AO systems have two symptoms: *code scattering* and *code tangling*. *Code scattering* means that the code that implements a crosscutting concern is spread across the system, and *code tangling* means that the code that implements some concern is mixed with code from other (crosscutting) concerns.

1.3. **Related Work.** Many aspect mining techniques have been proposed so far ([2], [5], [12], [13], [15], [16], [17]). [5], [13] and [16] use clustering for identifying crosscutting concerns, but in different contexts.

In [13] we have proposed a clustering approach in aspect mining based on *k-means* and *hierarchical agglomerative* clustering. We have also defined in [14] a set of new quality measures in order to evaluate the results of clustering based aspect mining techniques. Based on the approach proposed in [13], this paper presents a comparison of three clustering algorithms: *k-means*, *fuzzy c-means* and

hierarchical agglomerative, both from the aspect mining and clustering points of view.

The paper is structured as follows. Section 2 presents the context in which clustering is used in aspect mining. The clustering algorithms used in our comparison are described in Section 3. A comparative analysis of the results obtained on two case studies, based on some quality measures, is reported in Section 4. Section 5 presents some conclusions and further work.

2. Clustering approach in the context of aspect mining

In this section we present the problem of identifying *crosscutting concerns* as a clustering problem.

2.1. **Formal model.** Let $M = \{m_1, m_2, ..., m_n\}$ be the software system, where $m_i, 1 \leq i \leq n$ is a method of the system. We denote by n ($|M|$) the number of methods in the system.

We consider a crosscutting concern as a set of methods $C = \{c_1, c_2, ..., c_{cn}\}$, methods that implement this concern. The number of methods in the crosscutting concern C is $cn = |C|$. Let $CCC = \{C_1, C_2, ..., C_q\}$ be the set of all crosscutting concerns that exist in the system M. The number of crosscutting concerns in the system M is $q = |CCC|$.

Partition of a system M.
The set $\mathcal{K} = \{K_1, K_2, ..., K_p\}$ is called a **partition** of the system M iff $1 \leq p \leq n$, $K_i \subseteq M, K_i \neq \emptyset, \forall i \in \{1, 2, ..., p\}$, $M = \bigcup_{i=1}^{p} K_i$ and $K_i \cap K_j = \emptyset$, $\forall i, j \in \{1, 2, ..., p\}, i \neq j$.

In the following we will refer K_i as the i-th *cluster* of $\mathcal{K}$ and $\mathcal{K}$ as a *set of clusters*.

In fact, the problem of aspect mining can be viewed as the problem of finding a partition K of the system M.

2.2. **Identification of crosscutting concerns.** The steps for identifying the crosscutting concerns are as follows:

- **Computation** -Computation of the set of methods in the selected source code, and computation of the attributes set values, for each method in the set.
- **Filtering** - Methods belonging to some data structures classes like *ArrayList, Vector* are eliminated. We also eliminate the methods belonging to some built-in classes like *String, StringBuffer, StringBuilder, etc.*
- **Grouping** - The remaining set of methods is grouped into clusters using a clustering algorithm (*KM*, *FCM* or *HAC*, in this paper). The clusters

are sorted by the average distance from the point 0_l in descending order, where 0_l is the l dimensional vector with each component 0.

- **Analysis** - The clusters obtained are analyzed to discover which clusters contain methods belonging to crosscutting concerns. We analyze the clusters whose distance from 0_l point is greater than a threshold (eg. two).

3. Clustering Algorithms in Aspect Mining

In this section we briefly describe three clustering algorithms that we will use in the **grouping** step described in subsection 2.2, in order to identify a partition $\mathcal{K}$ of a system M.

In our approach, the objects to be clustered are the methods from the system $M = \{m_1, m_2, ...m_n\}$. The methods belong to the application classes or are called from the application classes.

We will consider each method as a l-dimensional vector: $m_i = (m_{i1}, \ldots, m_{il})$.

In our approach we have considered two vector-space models:

- The vector associated with the method m is $\{FIV, CC\}$, where FIV is the fan-in value and CC is the number of calling classes. We denote this model by $\mathcal{M}_1$.
- The vector associated with the method m is $\{FIV, B_1,\ B_2, ...B_{l-1}\}$, where FIV is the fan-in value and B_i is the value of the attribute corresponding to the application class C_i. The value of B_i is 1, if the method M is called from a method belonging to C_i, and 0, otherwise. We denote this model by $\mathcal{M}_2$.

3.1. Hard k-means clustering (KM).

Hard k-means clustering is also known as *c-means* clustering. The *k-means* algorithm partitions the collection of n methods of the system M into k distinct and non-empty clusters. The partitioning process is iterative; it stops when a partition that minimizes the squared sum error (*SSE*) is achieved. The *SSE* of a partition $\mathcal{K}$ is defined as:

$$SSE(\mathcal{K}) = \sum_{j=1}^{p} \sum_{m_i^j \in K_j} d^2(m_i^j, f_j) \tag{1}$$

where the cluster K_j is a set of methods $\{m_1^j, m_2^j, ..., m_{n_j}^j\}$ and f_j is the centroid (mean) of K_j:

$$f_j = \left(\frac{\sum_{k=1}^{n_j} m_{k1}^j}{n_j}, \ldots, \frac{\sum_{k=1}^{n_j} m_{kl}^j}{n_j} \right)$$

Hence, the *k-means* algorithm minimizes the intra-cluster distance. The algorithm starts with k initial centroids, then iteratively recalculates the clusters

(each object is assigned to the closest cluster - centroid) and their centroids until convergence is achieved.

The main disadvantages of *k-means* are:

- The performance of the algorithm depends on the initial centroids. So the algorithm gives no guarantee for an optimal solution, corresponding to the global objective function minimum.
- The user needs to specify the number of clusters in advance.

In order to avoid these two main disadvantages of *k-means*, based on the approach presented in [13], we propose a new heuristic for choosing the number of clusters and the initial centroids. This heuristic will provide a good enough selection for the initial centroids.

We use the following *heuristic* for choosing the number of clusters and the initial centroids:

(i) The initial number k of clusters is n (the number of methods from the system).
(ii) The method chosen as the first centroid is the most "distant" method (the method that maximizes the sum of distances from the other methods).
(iii) The next centroid is chosen as the method that is the most distant from the nearest centroid already chosen, and this distance is strictly positive. If such a method does not exist, the number k of clusters will be decreased.
(iv) The step (iii) will be repeatedly performed, until k centroids will be reached.

3.2. **Fuzzy C-means Clustering (FCM).** *Fuzzy c-means clustering* ([1], [6]), also known as Fuzzy ISODATA, is a clustering technique which is separated from hard k-means that employs hard partitioning. *FCM* employs fuzzy partitioning such that a data point (method) can belong to all groups with different membership degrees between 0 and 1.

FCM is representative for the method of *overlapping* clustering. It uses fuzzy sets to cluster data, so each point may belong to two or more clusters with different degrees of membership. In this case, data will be associated to an appropriate membership value.

We will denote by k the number of clusters that we want to obtain in the data set, that was determined by applying *KM*. A membership matrix U is used, so that the equality below holds.

$$\sum_{i=1}^{k} U_{ij} = 1, \forall j \in \{1, 2, ..., n\}$$

In the above equation, U_{ij} $(i \in \{1, , 2..., k\}, j \in \{1, , 2..., n\})$ represents the membership degree of method j to cluster i.

By iteratively updating the cluster centers and the membership degrees for each method [1], *FCM* iteratively moves the cluster centers to the "right" location within the data set.

FCM does not ensure that it converges to an optimal solution, because the initial centroids (the initial values for matrix U) are randomly initialized.

FCM reports the final values for the matrix U. We propose the following equation:

$$K_i = \{j \mid j \in \{1, 2, ..., n\} \text{ and } U_{ij} > U_{rj}, \forall r \in \{1, 2, ..., n\}, r \neq j\},$$

in order to identify the clusters in data, after *FCM* was applied.

We mention that the number of clusters reported by *FCM* is less or equal to k (there is a possibility to obtain empty clusters).

3.3. **Hierarchical Agglomerative Clustering (HAC).** The agglomerative (bottom-up) clustering methods begin with n singletons (sets with one element), merging them until a single cluster is obtained. At each step, the most similar two clusters are chosen for merging.

With the optimal number of clusters k determined after applying *KM*, we have applied a modified version of the traditional *HAC* algorithm in order to determine k clusters in data (the agglomerative algorithm stops when k clusters are reached).

4. Experimental Evaluation

In order to evaluate the results of the proposed clustering algorithms, we consider two case studies that are briefly described in Subsection 4.2. The obtained results are evaluated using three quality measures that are defined in Subsection 4.1.

4.1. **Quality Measures.** In this section we propose quality measures for evaluating the results of clustering based aspect mining techniques. The first measure (*SSE*) evaluates a partition from the clustering point of view, and the last two measures (*PAM*, *ACC*) evaluate a partition from the aspect mining point of view. In the following, we denote by $|A|$ the cardinality of the set A.

Squared Sum Error of a partition - *SSE*. The *squared sum error* of a partition $\mathcal{K}$, denoted by *SSE(K)*, is defined as in equation (1).

From the point of view of a clustering technique, smaller values for SSE indicate better partitions, meaning that SSE has to be minimized.

Percentage of analyzed methods for a partition - PAM [14]**.**

Let us consider that the partition $\mathcal{K}$ is analyzed in the following order: $K_1, K_2, ..., K_p$.

The percentage of analyzed methods for a partition $\mathcal{K}$ with respect to the set CCC, denoted by $PAM(CCC, \mathcal{K})$, is defined as:

$$PAM(CCC, \mathcal{K}) = \frac{\sum_{i=1}^{q} pam(C_i, \mathcal{K})}{q}.$$

$pam(C, \mathcal{K})$ is the minimum percentage of the methods that need to be analyzed in the partition $\mathcal{K}$ to discover the crosscutting concern C and is defined as:

$$pam(C, \mathcal{K}) = \frac{\sum_{j=1}^{i} |K_j|}{|M|}$$

where $i = min\{t \,|\, 1 \leq t \leq p \text{ and } K_t \cap C \neq \emptyset\}$ is the index of the first cluster in the partition $\mathcal{K}$ that contains methods from C.

$PAM(CCC, \mathcal{K})$ defines the percentage of the minimum number of methods that need to be analyzed in the partition in order to discover all crosscutting concern that are in the system M. We consider that a crosscutting concern was discovered the first time a method that implements it was analyzed.

Based on the definition, $PAM(CCC, \mathcal{K}) \in (0, 1]$. If each $C \in CCC$ has one method in the first analyzed cluster K_1 of the partition $\mathcal{K}$, then $PAM(CCC, \mathcal{K}) = \frac{|K_1|}{|M|}$, otherwise $PAM(CCC, \mathcal{K}) > \frac{|K_1|}{|M|}$.

Smaller values for PAM indicate short time for analysis, meaning that PAM has to be minimized.

Accuracy of a clustering based aspect mining technique - ACC. Let $\mathcal{T}$ be a clustering based aspect mining technique.

The accuracy of $\mathcal{T}$ with respect to a partition $\mathcal{K}$ and the set CCC, denoted by $ACC(CCC, \mathcal{K}, \mathcal{T})$, is defined as:

$$ACC(CCC, \mathcal{K}, \mathcal{T}) = \frac{\sum_{i=1}^{q} acc(C_i, \mathcal{K}, \mathcal{T})}{q}.$$

$$acc(C, \mathcal{K}, \mathcal{T}) = \begin{cases} \frac{|C \cap K_j|}{|C|} & \text{, if } K_j \text{ is the first cluster in which } C \text{ was discovered by } \mathcal{T} \\ 0 & \text{, otherwise} \end{cases}$$

is the accuracy of $\mathcal{T}$ with respect to the crosscutting concern C. For a given crosscutting concern $C \in CCC$, $acc(C, \mathcal{K}, \mathcal{T})$ defines the proportion of methods from C that appear in the first cluster where C was discovered.

In all clustering based aspect mining techniques, only a part of the clusters are analyzed, meaning that some crosscutting concerns may be missed.

Based on the above definition, $ACC(CCC, \mathcal{K}, \mathcal{T}) \in [0, 1]$. $ACC(CCC, \mathcal{K}, \mathcal{T}) = 1$ iff $acc(C, \mathcal{K}, \mathcal{T}) = 1, \forall\ C \in CCC$. In all other situations, $ACC(CCC, \mathcal{K}, \mathcal{T}) < 1$.

Larger values for ACC indicate better partitions with respect to CCC, meaning that ACC has to be maximized.

4.2. **Case Studies.** In order to evaluate the results, we consider two case studies: Carla Laffra's implementation of Dijkstra algorithm [10] and JHotDraw, version 5.2 [8].

The first case study is a Java applet that implements Dijkstra algorithm in order to determine the shortest path in a graph. It was developed by Carla Laffra and consists in **6** classes and **153** methods.

The second case study is a Java GUI framework for technical and structured graphics, developed by Erich Gamma and Thomas Eggenschwiler, as a design exercise for using design patterns. It consists in **190** classes and **1963** methods.

4.3. **Comparative Analysis of the Results.** In this section we comparatively present the results obtained after applying the clustering methods described in Section 3 with respect to the quality measure described above, for the case studies presented in this section.

Case study	Clustering algorithm	Model	No. of clusters	PAM	ACC	SSE
Laffra	KM	$\mathcal{M}_1$	13	0.0964	0.6666	0
Laffra	KM	$\mathcal{M}_2$	22	0.1029	0.6666	0
Laffra	FCM	$\mathcal{M}_1$	11	0.0947	0.6666	2.9148
Laffra	FCM	$\mathcal{M}_2$	19	0.0996	0.6666	7.0921
Laffra	HAC	$\mathcal{M}_1$	13	0.1241	0.5000	18.8685
Laffra	HAC	$\mathcal{M}_2$	22	0.1307	0.5000	16.7567
JHotDraw	KM	$\mathcal{M}_1$	93	0.0736	0.2782	0
JHotDraw	KM	$\mathcal{M}_2$	586	0.0770	0.2782	0
JHotDraw	FCM	$\mathcal{M}_1$	89	0.0735	0.2782	10.9971
JHotDraw	FCM	$\mathcal{M}_2$	26	0.0342	0.7812	64084.92
JHotDraw	HAC	$\mathcal{M}_1$	93	0.0718	0.3095	267.3225
JHotDraw	HAC	$\mathcal{M}_2$	586	0.0811	0.2782	119.6836

TABLE 1. The values of the quality measures for the two case studies.

Table 1 presents the results obtained by applying the three clustering algorithms, for the two vector space models.

As presented in Subsection 4.1, the partitions that minimize the *squared sum error (SSE)* are better from the clustering point of view.

The conclusions reached after analyzing the obtained results from the aspect mining point of view, are presented below.

The analysis of the results based on the clustering algorithm used:

Laffra case study

- For all algorithms, vector space model $\mathcal{M}_1$ has provided better results.
- The best results were obtained using *FCM*.

JHotDraw case study

- Vector space model $\mathcal{M}_1$ has provided better results for *KM* and *HAC*, and vector space model $\mathcal{M}_2$ for *FCM*.
- The best results were obtained using *FCM*.

The analysis of the results based on the vector space model used:

Laffra case study

- For both vector space models, *FCM* has provided better results.

JHotDraw case study

- For vector space model $\mathcal{M}_1$, *HAC* has provided better results, and for vector space model $\mathcal{M}_2$, *FCM* has provided better results.

After analyzing the obtained results, we can conclude that the vector space models used in the proposed clustering based aspect mining technique should be improved. This can also be the cause of the lack of correlation between the results from the aspect mining point of view and from the clustering point of view.

5. Conclusions and Future Work

In this paper we have comparatively presented the results of applying three clustering algorithms in aspect mining. The comparison was made mostly from the aspect mining point of view, using a set of quality measures (*SSE*, *PAM*, *ACC*).

Further work can be done in the following directions:

- To improve the vector-space models used in this clustering based aspect mining approach. In our opinion, the vector space models have significantly influenced the obtained partitions.
- To compare, from the aspect mining point of view, the results obtained by the clustering algorithms proposed in this paper with other clustering approaches that were proposed in the literature (such as variable selection for hierarchical clustering [3], search based clustering [11]).
- To apply this approach for other case studies like PetStore and JEdit.

References

[1] S. Albayrak and F. Amasyali. Fuzzy c-means clustering on medical diagnostic systems. In *Turkish Symposium on Artificial Intelligence and Neural Networks - TAINN*, 2003.

[2] S. Breu and J. Krinke. Aspect Mining Using Event Traces. In *Proc. International Conference on Automated Software Engineering (ASE)*, pages 310–315, 2004.

[3] E. B. Fowlkes, G. Gnanadesikan, and J. R. Kettering. *Design, Data, and Analysis: By Some Friends of Cuthbert Daniel*. Wiley, New York, NY, 1987.

[4] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, 2001.
[5] L. He and H. Bai. Aspect Mining using Clustering and Association Rule Method. *International Journal of Computer Science and Network Security*, 6(2A):247–251, February 2006.
[6] A. Jain and R. Dubes. *Algorithms for Clustering Data*. Prentice Hall, Englewood Cliffs, New Jersey, 1998.
[7] A. Jain, M. N. Murty, and P. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, 1999.
[8] JHotDraw Project. http://sourceforge.net/projects/jhotdraw.
[9] G. Kiczales, J. Lamping, A. Menhdhekar, C. Maeda, C. Lopes, J.-M. Loingtier, and J. Irwin. Aspect-Oriented Programming. In *Proceedings European Conference on Object-Oriented Programming*, volume 1241, pages 220–242. Springer-Verlag, 1997.
[10] C. Laffra. Dijkstra's Shortest Path Algorithm. http://carbon.cudenver.edu/ hgreenbe/ courses/dijkstra/DijkstraApplet.html.
[11] S. Mancoridis, B. S. Mitchell, Y. Chen, and E. R. Gansner. Bunch: A Clustering Tool for the Recovery and Maintenance of Software System Structures. In *ICSM '99: Proceedings of the IEEE International Conference on Software Maintenance*, pages 50–59. IEEE Computer Society, 1999.
[12] M. Marin, A. van, Deursen, and L. Moonen. Identifying Aspects Using Fan-in Analysis. In *Proceedings of the 11th Working Conference on Reverse Engineering (WCRE2004)*, pages 132–141. IEEE Computer Society, 2004.
[13] G. S. Moldovan and G. Serban. Aspect Mining using a Vector-Space Model Based Clustering Approach. In *Proceedings of Linking Aspect Technology and Evolution Workshop(LATE 2006)*, Bonn, Germany, March 2006.
[14] G. S. Moldovan and G. Serban. Quality Measures for Evaluating the Results of Clustering Based Aspect Mining Techniques. In *Proceedings of Towards Evaluation of Aspect Mining(TEAM), ECOOP*, 2006, to be published.
[15] Orlando Alejo Mendez Morales. Aspect Mining Using Clone Detection. Master's thesis, Delft University of Technology, The Netherlands, August 2004.
[16] D. Shepherd and L. Pollock. Interfaces, Aspects, and Views. In *Proceedings of Linking Aspect Technology and Evolution Workshop(LATE 2005)*, March 2005.
[17] P. Tonella and M. Ceccato. Aspect Mining through the Formal Concept Analysis of Execution Traces. In *Proceedings of the IEEE Eleventh Working Conference on Reverse Engineering (WCRE 2004)*, pages 112–121, November 2004.

Babeş-Bolyai University, Faculty of Mathematics and Computer Science, Cluj-Napoca, Romania
E-mail address: gabis@cs.ubbcluj.ro

Babeş-Bolyai University, Faculty of Mathematics and Computer Science, Cluj-Napoca, Romania
E-mail address: grigo@cs.ubbcluj.ro