Requirements Interaction Management Clustering

Mona Assarandarban
Department of Electrical Engineering and Computing Systems
University of Cincinnati
Cincinnati, USA
assarama@mail.uc.edu

Nan Niu
Department of Electrical Engineering and Computing Systems
University of Cincinnati
Cincinnati, USA
niunn@ucmail.uc.edu

Abstract— **There is a strong need to automate the process of Requirements Interaction Management (RIM) since the number of requirements for a system that need to be managed is increasing considerably. In this paper, we propose a method for making requirements interaction management automatic. To do so, we took advantage of information retrieval methods to gain useful information out of raw requirements which was accordingly used by clustering methods to find the structure of interactions between requirements. The results show that our approach can deliver stable clusters of requirements that interact similarly.**

***Keywords*—Requirements interaction management; term frequency-inverse document frequency; similarity; clustering**

I. Introduction

In this day and age, finding significant relationships between requirements, managing and arranging them in an appropriate way has changed to one of popular topics in the area of requirements engineering. Traditionally the process of discovering requirements interactions was performed manually; however, discovering large number of requirements along with managing their interactions require automated interaction processing systems.

There are some precious research that focuses on clustering requirements based on various features. Some proposed to uncover clusters of requirements through discovering essential attributes [1][2]. The others paid attention to construction of feature models based on requirements clustering [3]. It is so while, very few works concentrated on adaptation of automated techniques for interactions management. This paper introduces a method for generally analyzing the requirements interaction of a library system using clustering.

The purpose of proposed system is to improve the interactions between requirements through requirements clustering. Resource is considered as the basis of requirements interaction in which two requirements depend on the same resource. We initially found critical similarities between requirements. Next, we looked for attributes that played important role in determining relationship between requirements that interact through similar resource. Finally, we applied clustering methods on the data based on obtained features.

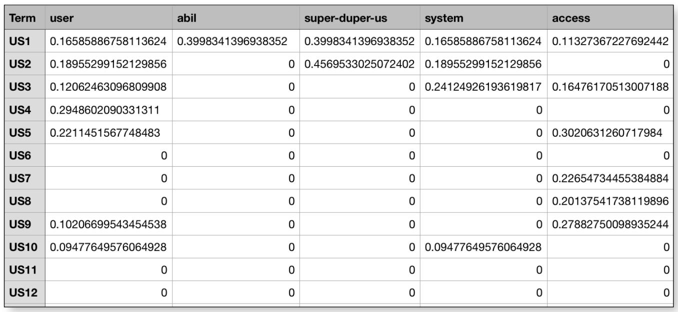
II. Dataset

In our dataset, we took advantage of user stories that were defined for the library system of University of Cincinnati. The dataset includes forty seven requirements which are the corresponding user stories. First, we calculated the term frequency-inverse document frequency values of all existed terms in requirements for all user stories. In the same way, we removed stop words and applied a stemming algorithm to obtain the root of our English terms, using Porter's algorithm [4]. Table 1 shows the tf-idf values of a number of terms for some requirements. Following that, two specific resources, metadata and workgroup, are used for discovering interactions. At this point, there are two cases for shrinking the dataset. First, the dataset can be reduced to requirements interacting via the two resources, requirement filtering (see Table 2). Second, useful terms in resource interactions can be taken into consideration for decreasing the number of attributes for dataset, term filtering (see Table 3).

III. Methods

After preparing dataset, clustering methods are employed in order to group interacting requirements based on resource. In other words, we used some unsupervised learning methods for clustering requirements that are interacting via the same resource into one group. As a result, we ended up with two clusters indicating metadata and workgroup resources, such that each cluster included requirements that interact via

Table 1: *tf-idf values of terms for some requirements*



Term	user	abil	super-duper-us	system	access
US1	0.16585886758113624	0.3998341396938352	0.3998341396938352	0.16585886758113624	0.11327367227692442
US2	0.18955299152129856	0	0.4569533025072402	0.18955299152129856	0
US3	0.12062463096809908	0	0	0.24124926193619817	0.16476170513007188
US4	0.2948602090331311	0	0	0	0
US5	0.2211451567748483	0	0	0	0.3020631260717984
US6	0	0	0	0	0
US7	0	0	0	0	0.22654734455384884
US8	0	0	0	0	0.20137541738119896
US9	0.10206699543454538	0	0	0	0.27882750098935244
US10	0.09477649576064928	0	0	0.09477649576064928	0
US11	0	0	0	0	0
US12	0	0	0	0	0

Table 2: *Requirement filtering*

Term	user	abil	superduperus	system	access	includ	view	restrict	content
US4	0.2993496995481354	0	0	0	0	0	0.2558427881104495	0	0
US9	0.10362104984358535	0	0	0	0.2819356098074324	0	0	0	0.1958408732019827
US11	0	0	0	0	0	0	0	0	0.1414406306458764
US12	0	0	0	0	0	0	0	0	0
US13	0	0	0	0	0	0	0	0	0.12729656758128877
US14	0	0	0	0	0	0	0	0	0
US15	0	0	0	0	0	0	0	0	0
US16	0.10362104984358535	0	0	0	0.2819356098074324	0	0	0	0.1958408732019827
US17	0	0	0	0	0	0	0	0	0
US18	0	0	0	0	0	0	0	0	0.15912070947661094
US19	0	0	0	0.11225613733055079	0	0	0	0	0.10608047298440729
US20	0	0	0	0	0	0	0	0	0
US21	0	0	0	0.09621954628332924	0	0	0	0	0
US22	0	0	0	0	0	0	0	0	0
US31	0	0	0	0.10362104984358535	0	0	0	0	0
US41	0	0	0	0	0	0	0	0	0
US42	0	0	0	0	0	0	0	0	0
US43	0	0	0	0.22451227466110157	0	0	0	0	0
US47	0	0	0	0	0	0	0	0	0.09792043660099135

Table 3: *Term filtering*

US	Workgroup	Metadata
US1	0	0
US2	0	0
US3	0	0
US4	0.23973548030487332	0
US5	0	0
US6	0	0
US7	0	0
US8	0	0
US9	0.08298535856707154	0
US10	0	0
US11	0.11986774015243666	0
US12	0.215761932274386	0
US13	0.215761932274386	0
US14	0.215761932274386	0
US15	0.11986774015243666	0
US16	0.08298535856707154	0
US17	0.13485120767149125	0.5300658840500228
US18	0.13485120767149125	0
US19	0.08990080511432749	0
US20	0.215761932274386	0.4240527072400182
US21	0.07705783295513785	0
US22	0.107880966137193	0
US23	0	0
US24	0	0
US25	0	0

one specific resource. Regarding clustering, two methods has been used for group requirements as following:

1. K-Means clustering:

The first method is k-means clustering since it is simple to implement and often works well. K-means clustering aims to partition all requirements into k clusters in a way that each requirement belongs to the cluster with the closest mean. Hence, each requirement is considered as a point in a 2 dimensional space with metadata tf-idf value for X axis and workgroup tf-idf value for Y axis. Figure 1 demonstrates the result of k-means clustering on our reduced dataset in which each color shows members of one cluster.

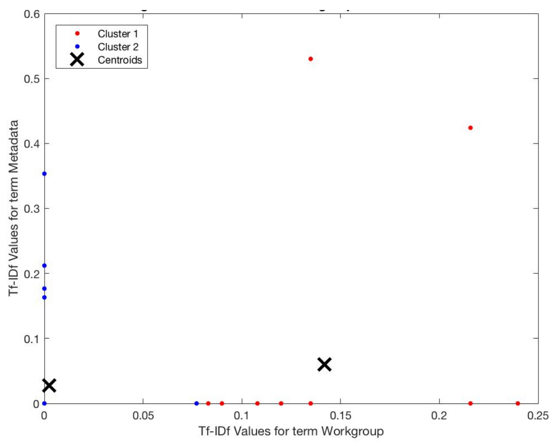


Figure 1: *K-means clustering of requirements based on workgroup and metadata resources*

2. Hierarchical clustering:

Hierarchical clustering is the second clustering method that we applied to our dataset, which builds a hierarchy of clusters [5]. Firstly, we calculated the similarity between requirements by cosine similarity measure. It can be described as a measure of similarity between any pairs of requirements such that the cosine of the angle between them is considered as the similarity. In our case, cosine similarity is one of the most effective measures since our requirements are considered as vectors. The corresponding similarity measure can be calculated by:

$$Similarity = A\,.\,B / |A|^2 |B|^2 \qquad (1)$$

Then hierarchical clustering grouped the requirements into clusters based on cosine similarity values. By way of illustration, each requirement is considered as one cluster and then the most or least similar pairs of clusters are merged to form new clusters at each iteration. Single linkage clustering is one type of hierarchical clustering which not only focuses on most similar observations but also produces the best result among all other hierarchical clustering types and is defined with the following equation:

$$sim(c_i, c_j) = \max_{x \in c_i, y \in c_j} sim(x, y) \qquad (2)$$

Therefore, we used the corresponding method to cluster our requirements. In addition, hierarchical clustering is reliable in terms of prediction. This type of flat clustering is called to be deterministic since the results are more predictable compared to k-means. Figure 2 shows the result of hierarchical clustering on our dataset.

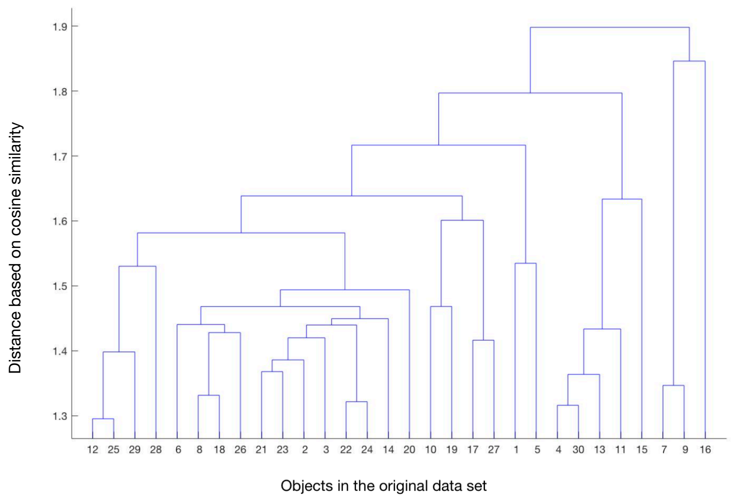


Figure 2: *Binary cluster tree created by single linkage function*

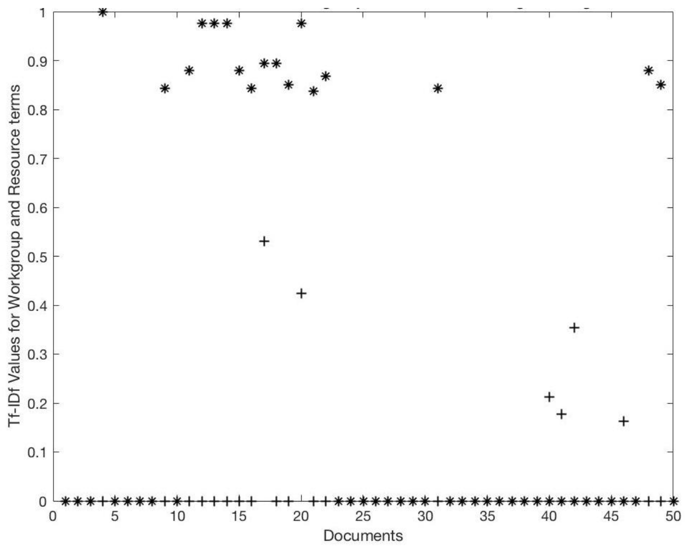


Figure 3: *Increasing tf-Idf weights for the workgroup resource; the points which are plotted by + are the requirements that interact via metadata resource; on the other hand, * points refer to requirements interacting through workgroup. After increasing weights of requirements for workgroup resource, the elements of two resources became completely separated. So, a centroid-based clustering method like k-means can define clusters with high accuracy.*

IV. Results

When it comes to results, we can infer that k-means clustering produces better results for our dataset compared to hierarchical clustering. Although the number of incorrectly clustered instances in k-means is less than hierarchical clustering, there is still some error in k-means results. In order to improve the results, we can increase the tf-idf weights for resources which are important at a time. For instance, if obtaining the requirements that are interacting through workgroup resource is the main focus, the tf-idf values of the term workgroup is increased. We used the weight update formula of perceptron in neural network [6] to change the tf-idf weights as:

$$w = w + n(y - y')x \tag{3}$$

We can describe n as learning rule which is a small value; y as output or class labels; x as input. Since in unsupervised learning we do not have any response for instances, we remove y in calculations. Figure 3 shows how increased tf-idf weights has separated the requirements interacting through different resources.

Following that, we applied k-means clustering on new dataset with updated tf-idf values and obtained less error. As Figure 4 illustrates, all requirements are correctly assigned to their resource cluster. The requirements which are interacting via both resources are assigned to the cluster of resource which has higher importance in terms of increased tf-idf values.

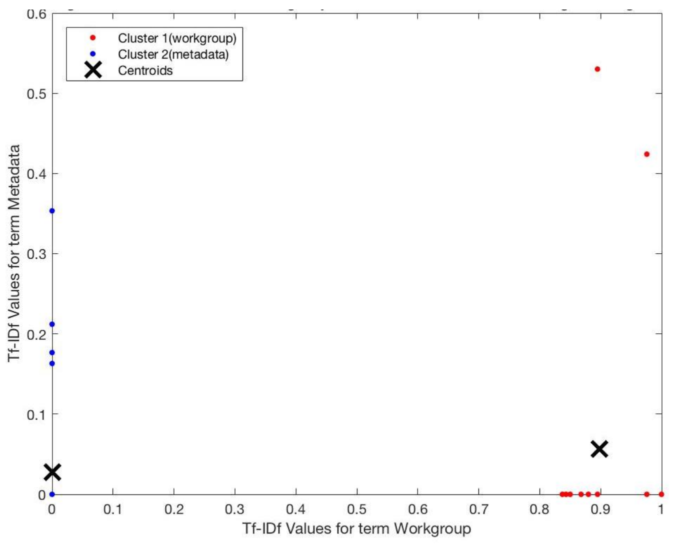


Figure 4: *K-means clustering with higher weights for workgroup resource*

V. CONCLUSION

In this paper, we introduced an approach for requirements interaction management clustering with purpose of automating the identification of requirements that interact with each other. The main goal is clustering requirements for which there is relationship between them based on a specific resource. Two different clustering methods have been used to group requirements by means of two values: term frequency-inverse document frequency and similarity measures. Regarding dataset, we took advantage of user stories for UC library system as our requirements, which was then followed by data preparation to obtain our dataset based on requirements and existed terms in their contents. Moreover, some other strategies has been proposed to improve the clustering results. Through prioritizing tf-idf values of requirements for a specific resource, we were able to increase our model performance in an effective way. The results demonstrate that k-means clustering performed well with our small dataset.

Our approach can also be effective in increasing the number of desired interactions as well as minimizing unwanted interactions. By the same token, our strategy can be used later to inject the former interactions to the latter ones in order to make them more efficient. Moreover, defining different clusters allow us to assign each developer a specific cluster to implement. Hence, analyzing smaller partitions of system can lead to better efficiency of the system.

The approach that we proposed is still in progress and needs more improvements. First and foremost, we plan to increase the size of training data to have a better training on dataset using clustering. In other words, we want to add all keywords that play important role in resource-based interactions among requirements. Furthermore, the problem can be expanded in terms of basis of interactions. We can increase the dimension of our problem by adding more interaction types to our dataset, such as structure, task, time, etc. Last but not least, discovering other clustering methods that works for our dataset is another solution for improving our system. D-IMPACT clustering is a data preprocessing algorithm which can improve the performance of RIM clustering. In D-IMPACT, data points are iteratively moved based on density and attraction in order to separate clusters [7]. Undoubtedly, we hope that our method positively contributes to improvement of RIM in Requirements Engineering.

REFERENCES

1. Duan, C., Dumitru, H., Cleland-Huang, J. and Mobasher, B., 2015, March. User-Constrained Clustering in Online Requirements Forums. In International Working Conference on Requirements Engineering: Foundation for Software Quality (pp. 284-299). Springer, Cham.
2. Niu, N. and Easterbrook, S., 2008, September. On-demand cluster analysis for product line functional requirements. In Software Product Line Conference, 2008. SPLC'08. 12th International (pp. 87-96). IEEE.
3. Chen, K., Zhang, W., Zhao, H. and Mei, H., 2005, August. An approach to constructing feature models based on requirements clustering. In Requirements Engineering, 2005. Proceedings. 13th IEEE International Conference on (pp. 31-40). IEEE.
4. https://tartarus.org/martin/PorterStemmer/
5. Johnson, S.C., 1967. Hierarchical clustering schemes. *Psychometrika*, *32*(3), pp.241-254.
6. Rosenblatt, F., 1958. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, *65*(6), p.386.
7. Wang, X., Qiu, W. and Zamar, R.H., 2007. CLUES: A non-parametric clustering method based on local shrinking. *Computational Statistics & Data Analysis*, *52*(1), pp. 286-298.