Using Requirements Clustering to Discover Dependent Requirements for Hidden Impact Analysis

Ahmed Safwat(✉) and Mostafa Mohamed Yacoub

Sadat Academy for Management Sciences, Faculty of Computers and Information, Cairo, Egypt
asafwat@sadatacademy.edu.eg

Abstract. Since one of the most important practices in any software development lifecycle is Requirements Engineering. Weakly applied RE is one of the main causes for the development breakdown. Main practices of RE processes are specification and elicitation, verification, cooperation, and implementation management. Since ULS systems are increasing in complexity and difficulty, it has observed a growing call for smart modules, methods, and applications that would support improving RE practices. In this paper, we focus on how various kinds of recommendation technologies can be applied to support ULS participants in the completion of the RE tasks; first, we provide an overview of the research related to the application of recommendation tools in RE that was already described in a high level. Second, we show in detail how clustering method as one of Model-based filtering technique can be applied to proactively strengthen "Measuring Change Ripple Effect", Third, new ideas need to be discussed and future research explored. We have used Natural Language Processing (NLP) and Similarity Models to support the model.

Keywords: ULS · ULS Challenges · Software Engineering · Requirements Engineering · NLP · Similarity Model · Change Management

1 Introduction

Change is inevitable in all large-scale projects. Changes are pushing the system to respond to external and internal requirements changes as long as the demands change. In previous work we were able to measure the impact of single or group of changes on ULS requirements repository using NLP similarity model. However, the existing requirements could be deleted or changed themselves without exposing to external change requests.

Therefore, in this paper, we propose a module to automatically cluster the existing requirements based on similarity measures. An empirical evaluation is managed using the same ecosystem ERP [1] dataset to evaluate our model. The experimental results showed that the proposed model specified semantic clustering according to k-means unsupervised clustering method.

R. Hou et al. (Eds.): BDTA 2021/2022, LNICST 480, pp. 3–23, 2023.
https://doi.org/10.1007/978-3-031-33614-0_1

Since one of the most important practices in any software development lifecycle is Requirements Engineering. Weakly applied RE is one of the main causes for the breakdown of the development. Main practices of RE processes are specification and elicitation, verification, cooperation, and implementation management. Since ULS systems are increasing in complexity and difficulty, it's observed a growing call for smart modules, methods, and applications that would support improving RE practices.

In this paper, we focus on how various kinds of recommendation technologies can be applied to support ULS participants in the completion of the RE tasks [2].

A recommender system can be defined as any system that guides a user in a personalized way according to his/her interest in useful objects in a large space, recommendation systems are useful for suggesting products or services such as books, movies, goods, and financial assistances. Such systems help the end customers in the selection of related items in circumstances where the volume of varieties exceed their ability to review it and to get a choice [3].

Low-movement items like movies and products are also suggested by observing customer's preferences for an equally scoring behavior. The related form of suggestion is called Collaborative Filtering [4], Which is a clear applichation of word-of-mouth ads where buying decisions are prejudiced by relationships and colleagues' views: In the event that two consumers viewed related objects in a similar manner in the long-ago, a mutual filtering-based recommendation framework will propose to one consumer new things that the other has already rated with confidence. For example, Netflix online movies platform recommends movies already viewed by customers with a similar ranking. [5].

A substitution tool for endorsing low-movement products is content-based filtering [6]. It is a filtering technique in which characters of items are used by a user ranked in the history to identify new suggestions for the same customer. For instance, where a customer of amazon.com has purchased Linux OS-related products, books (related to Linux) will be recommended for potential recommendation transactions. Fast-movement or related objects such as mobiles are endorsed based on knowledge-based recommendation methods, Where the Recommendation System uses predefined parameters for the identification of a selection of candidates [7]. Usually, ranking-built recommendation methods are not appropriate for slow-moving items since these items are not purchased routinely and thus no recent ranking data are provided.

Collaborative recommenders are one of the recommendation methods, which provides good proposals and recommendations. Collaborative filtering (CF) methods pay attention to the likeness and relationship among users' favorites. Therefore, associations among users are key instruments during the recommendation methods [8]. Recommender system methods and organization are given in Fig. 1.

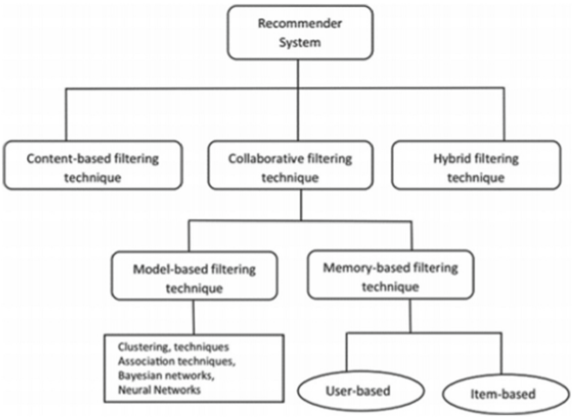

Fig. 1. A Summary of Methods of Recommender Systems Methods and Categorization

The major contributions of this paper are the following. First, we provide an overview of the research related to the application of recommendation tools in RE that was already described in a high level in Sect. 2.6. Second, we show in detail how clustering method as one of Model-based filtering technique can be applied to proactively strengthen "Measuring Change Ripple Effect" described paper 3 to support ULS technical participants in RE change management. Third, new ideas need to be discussed and future research explored (Table 1).

Table 1. Overview of Recommendation Methods in Supporting RE

RE Activity	Scenario	Recommended Module	Ref
Stakeholders Analysis	Recommending Stakeholders	Stake Source Using Social Networks and Crowdsourcing	Soo et al. [9]
Requirements Elicitation	Recommending Requirements	Data Mining and Recommendation Systems and k-Nearest Neighbor Algorithm: kNN detects like-minded users with similar rating history Content-based filtering Social network analysis Collaborative filtering	Niolofar et al. [10] Dumitru et al. [11] Soo et al. [12] Mobsher et al. [13]

(*continued*)

Table 1. (*continued*)

RE Activity	Scenario	Recommended Module	Ref
Requirements Prioritization	Prioritizing Requirements based on Stakeholders position and interests	StakeSource2, the ratings of stakeholders on requirements and their impact in the project The method uses data mining and machine learning to prioritize requirements fitting to stakeholders' interests, business areas, and concerns	Soo et al. [14] Chuan et al. [15]
Requirements Prediction	Predicting based on similar stakeholders' interest	Collaborative filtering to predict other requirements [12]	Soo et al. [14]
Quality Assurance	Finding Requirements Conflict Consistency Management Dependency detection Managing Feature requests	StakeSource2.0 which highlights stakeholders with conflicting preferences for requirements [14] Knowledge-based recommendation Clustering Clustering	Soo et al. [14] Felfernig [16] Cleland-Huang et al. [17]
Managing Requirements uncertainty	Specifying uncertainty within requirements model	Using MAVO to convey reduction of uncertainty in RE models	Rick et al. [18]

In the following section, we discuss existing research for requirements clustering in support of requirements planning and change management based on the collections of textual functional and non-functional requirements.

The basic concept is to evaluate the needs that are present in the requirements repositories and to apply clustering methods for the smart classes of these user requirements that can be evaluated in the future to identify hidden change impacts within similar clusters and for completeness control. The module proposed is to use k-means to detect the similarity between textual requirements. The module proposed is to use k-means to detect the similarity between textual requirements.

2 Requirements Management

The elicitation and interpretation of requirements emphasizes the collection of requirements to from different stakeholders. The elicitation and interpretation of requirements emphasizes the collection of requirements from different stakeholders. For example,

textual requirements specifications, business processes, use cases and prototypes and user interfaces are representative resulting artifacts [2].

Particularly in large-scale and spread software developments, it is unmanageable to establish individual conferences on a steady base. In cases like those, users Requirements are also outlined in online-based discussions that respond to the difficulties of information overwork, duplication, information imperfection and diverse participants' deviating thoughts. C-Huang et al. [13] and C-Herrera et al. [19], in their strategies to boost subsistence for participants in the production of ULS Systems, they illustrate the way of using clustering methods to categorize customer requirements and further on to allocate (recommended) participants to groups based on content-based filtering [6]. A major driver for such an allocation of stakeholders to requirement clusters is to achieve a succinct coverage, that is, an appropriate number of participants will evaluate and prove each requirement.

In contrast to their tactics of recommending (assigning) stakeholders to requirement clusters (topics), the tactics addressed in Cleland-Huang et al. [13] and Castro-Herrera et al. [19] often pay attention to supporting participants' requirements, focused, for example, on the ideas of collective filtering, based on the concepts of collaborative filtering for example, A big reason for implementing collaborative filtering in this scenario was the awareness of correlation effects that help to increase quality requirements (participants getting proposals concerning requirements they are concerned in have a high degree of similarity of evaluating their needs). Another reason for collaborative filtering presentation is to enhance the understanding of the requirement model when creating adapted navigation ways for participants.

Because requirements are expressed informally on a regular basis, usually relationships between requirements are illustrated in terms of conjunctions (e.g., requirement A requires requirement B or requirement A is incompatible with requirement B) defined by participants. Recommendation systems allow for the provision of extra information that proactively assists participants in defining requirement relationships.

The discovery of dependence between requirements can be based, for example, on clustering methods which are gathered into clusters of similar subjects (see, e.g., C-Huang et al. [17]). The basic idea is that needs which are assigned to the similar group depend on both sides. Though useful, this tactic isn't ended up to a broad description of the nature of reliance but then helps as a foundation for an additional investigation by stakeholders. Fantechi and Spinicci list some initial effort related to requirements and discovery of inconsistencies in "Open-Source Software Creation (OSSD)" based on the "Natural Language Processing (NLP)" techniques [20].

3 Requirements Clustering

There are different methods when there is a need to generate clusters from textual requirements collected by ULS stakeholders. Machine learning solutions those are expended most founded in for this resolution can be either "supervised" or "unsupervised". In supervised systems, classes are essentially to be predefined [21].

In supervised learning, the human experts who are involved to tag some amount of data with predefined classification, in that case the amount of data or requirements

are controlled by the knowledge workers. While Unsupervised learning algorithms permit the automatic specification of clusters without any earlier guidance supported by professionals [21].

Clustering in overall is a significant and valuable method that automatically classifies a group with a considerable number of data substances into a much lesser number of similar sets. In the specific situation of textual documents, clustering has verified to be an operational method and an exciting investigation issue as well. It is rising even more motivating and challenging with the growth of the internet and the development of Web 2.0. For example, outcomes resulted from search engines are clustered to aid end users rapidly classify and emphasis on the applicable set of outcomes. Client notes are grouped in numerous online stores, such as netflix.com, to deliver cooperative advice. In collaborative bookmarking or tagging, clusters of customers those have common specific characters are recognized by their behaviors [22].

Precise clustering needs an accurate description of the nearness between a couple of items, by either the likeness or distance. A diversity of likeness or distance methods have been offered and broadly affected, such as the Jaccard correlation coefficient and cosine similarity [23].

In the clustering method, we'll also have to distinguish the dissimilarity/similarity between two clusters or between a cluster and an entity.

3.1 Hierarchical Clustering

"Hierarchical clustering" algorithms recognized their term since they shape a set of clusters that may be represented as a hierarchy of clusters. The hierarchy would be built-in top to bottom (called divisive) or bottom to top (called agglomerative) style. Hierarchical clustering methods are one of the Distance based clustering processes, i.e., using a similarity method to calculate the intimacy among text documents.

In the top to bottom method, it starts with single cluster that contains all the text documents. it repetitively divides this cluster into sub-sets or sub-classes. In the agglomerative method, each document is originally reflected as a single cluster. Then sequentially the utmost alike clusters are combined till all documents are contained in one cluster.

Three diverse merger modules for agglomerative algorithms:

1) Single Linkage Clustering: In this method, the likeness among two groups of documents is the uppermost similarity among any couple of documents of these clusters.
2) Group-Average Linkage Clustering: In that type of clustering, the likeness between two clusters is the average likeness among couples of documents in these clusters.
3) Complete Linkage Clustering: within this technique, the likeness between two clusters is the poorest situation similarity between any couple of documents in those clusters [24].

Numerous algorithms are been projected for automatically clustering words founded on a huge corpus. They are divided into two categories., One category is built on rearranging words from group to group initially from some early set of classes. The second type repeats assembling classes beginning from a set of singleton classes (which contain only one word).

Both categories are share the same end function, in most conditions by confusion or average joint information. The value of the other category for the sake of building hierarchical clustering is that it could easily change the past of the merging method to a tree-structured forming of the vocabulary.

The other way around, the second category is disposed to be stuck by a local minimum.

The first category is stronger to the local minimum problem, but then the excellence of clusters importantly relies on the primary group of classes and discovery of an opening set of good quality is itself an identical problematic. Additionally, the first method only offers tools of dividing the vocabulary and it doesn't deliver a method of building a hierarchical clustering of words bags [25].

3.2 Clustering Using K-Means

K-means clustering [26] is an important technique for identifying clusters, where k states the number of clusters obtained. K requirements may be chosen as cluster cores in the preliminary repetition, and the other requirements are assigned to the closet cluster most.

Table 2. Example of System Requirements

Requirement	Category	Description
R1	Database	Indexed Configuration in DB
R2	User Interface	UI with internet help available
R3	Database	Isolated layer for DB
R4	User Interface	User Interface with enterprise identity

Table 3. Tokens extracted from System Requirements

Requirement	Extracted Tokens
R1	Indexed Configuration DB
R2	UI internet help
R3	Isolated layer DB
R4	User Interface enterprise identity

The likeness of the pull-out tokens – Eq. 1 reveals a basic clear resemblance metric for example, sim (r1, r3). 0.17, if we undertake tokens (r1). {indexed, configuration, DB} and tokens(r3). {isolated, layer, DB}:

$$\text{Sim(s, r)} = \frac{|tokens(s) \cap tokens(r)|}{|tokens(s) \cup tokens(r)|} \tag{1}$$

Different matching scoring could be implemented – For the sake of this case, we ask that the tokens be similar (see Table 3) mined from the written statements of our

sample requirements ({r1; r2; r3; r4} in Table 2). The centroid (mean) per cluster is then defined for each cluster and requirement specifications are allocated to clusters. For that case, both clusters c1:{r1; r3} and c2:{r2; r4} are expected to be recognized after one step where sim(r1; r3) is 0.17 and sim (r2; r4) is0.5. The algorithm ends if a significant reiteration depth is reached or all clusters remain steady [2].

So, the centroid's value in individual dimension is the calculated mean of that dimension above all the entities in the cluster. Let C be a set of documents. Its centroid is defined as

$$\overrightarrow{tc}. = \frac{1}{|C|} \sum_{\overrightarrow{td} \in C} \overrightarrow{td} \tag{2}$$

where it is the mean value of all vectors' terms in the group [27].

4 Determining the Good Number of Clusters

The clustering of requirements, though, is considerably harder than the clustering of normal documents in several behaviors. The cluster granularity, i.e., the exact value of clusters count is one of main issues that needs to be specified automatically in requirements clustering at a very acceptable level of granularity. While document clustering originates from the need to sort or filter huge groups of textual materials, such as journals, books, feeds, or internet pages, the resolution, that is frequently the main driver, of data clustering, is to classify the documents into a controlled number of classes to simplify some basic activities such as surfing and searching. The amount of the classes is naturally minor and is typically recognized beforehand. Instead, the objectives for clustering of requirements are very dynamic and are reliant on the activities for which the produced clusters will be use [28].

One of the basic problems in k-means methods is determining the good value of clusters. The right choice of k is often unclear; various experts used different methods to solve this problem, Elbow method is one of them to get the good number of K for K-mean algorithms.

Cross-validation would be extra method for inspecting the good value of clusters planned by Smyth [29], depending on cluster solidity to find out the correct k.

Another technique based on the average silhouette of the data is useful and reliable to identify the number of clusters accepted. The silhouette of a data instance is a degree of how narrowly it is ties to data point inside its own cluster and how roughly it is alike to data of the near cluster [30].

Intra and Inter Cluster Distances, the main objective after implementation of the clustering algorithm is to get a shortened and well-separated subsequent clustering. As a result, the intra-cluster distance (within cluster scatter distance) has to be minimized and the inter-cluster distance has to be maximized (between cluster separations) [31].

Hence, cluster density in K-means can be estimated using the intra-cluster distance dimension average. Inter-cluster distance is the distance between centers that shows separation of clusters, which must be amplified. The combination of these two measures (Elbow and Silhouette) can help calculate the fineness of the clustering as described by Siddheswar et al. [32]. Validity = intra/inter.

Given the fact that there is no adequate solution to the difficulties of measuring true K value estimation, which is accurate in each dimension, but some heuristic rules are used to calculate K value, (Intra and Inter-cluster distances) with k-mean++ allow a sensible choice of the preliminary centroid and it is significantly quicker and can be used in many ways to achieve the end result. [30].

4.1 Suggested Model for Calculating Good Number of K Clusters

It's not easy to divide a set of documents D into separate partitions or clusters and solid procedure since it requires to shadow a methodical process to cluster text data. The phases are utilized to highly conclude the clustering outcomes in the entire document clustering methods are examined in below:

Preprocessing
Preprocessing is the main obligatory step to clear up the data for text mining processes. It is useful for reducing data noise and for cleaning the data. As far as preprocessing is concerned, the actual purpose is to transform the originated data into a logical machine type. Preprocessing process requires tokenization, filtering, stemming or the elimination of stop-words.

Filtering Words
The terms that give low value below vector methods must be removed before the actual calculation; filtering is the way to achieve this mission. In addition to several documents, each document includes numerous words such as punctuations, special characters, stop words and redundant words. It provides imperfect information to distinguish between several documents, although it also contains infrequent words which offer no meaning and requests to get out [30].

Stemming
The fundamental goal of the stemming method is to adapt the words to their source (root/stem) words, which are extremely activity-based language. The method respects immensely the stemming method in English. It is a procedure that serving to increase the effectiveness and decline duplications [33].

Stop-Word Removal
"Stop words are the continuous words that occur, such as prepositions, articles, conjunctions: is, an, a, when, but etc., or terms that are not important and some terms of high frequency. While Stop-word elimination is the method used to dismiss these words from the vocabulary as a vector space perspective they do not confer any value and measure less important. Stop-word elimination procedure assists in implementation and significantly impacts the entire clustering process [34].

Processing
Like the K-Means and K-Medoids, the centroid-based cluster operators make a centroid model and a categorized set of clusters. The model of the centroid cluster has details about the clustering achieved. It states which cases are part of which cluster. It also contains

details on the centroids of each cluster. The Cluster Distance Performance operator returns this centroid cluster model and clustered collection as input and evaluates model output by cluster centroids. There are two output algorithms provided: Cluster distance average and Davies-Bouldin index [35].

Average Within Cluster Distance:
The average distance within a cluster is calculated by measuring the mean distance between the centroid and all points in a cluster [36].

Davies-Bouldin Index:
It is good to know that index (DB) is based on the fact that the inter-cluster separation and intra-cluster similarity and solidity should be high for a good partition. Then, the dispersion measure and the measure of cluster similarity must be defined to determine the DB index.

Since the goal is to achieve the lowest scattering within the cluster and the highest segregation between the clusters, the number of clusters c that minimizes VDB is considered the best value of c [37].

Representation
The important requirement is to transform our corpus into machine-detectable form vectors, described as VSM (Vector Space Model) before clustering the documents into sets $z = \{x_1, x_2, x_3, \ldots, x_n\}$. The term x_1 in Z represents the item vector d for each text document while $d = \{w_1, w_2, w_3, \ldots, w_n\}$. The w_i is a term weight demonstration of term t_i in a document that results in the impact of each word in a text. It is highly reliable and easier to make logical use of the TFIDF methodology for SVM corpus vectorization (Term Frequency Inverse Document Frequency) which is extremely utilized one, that calculates a position within the corpus for each term of a text. That calculates a position within the corpus for each term of a text. The TFIDF can be computed as follows:

$$w_{ji} = tf_{ji} * idf_{ji} = tf * log_2(\frac{n}{df_{ji}}) \tag{3}$$

The tf_{ji} is the occurrence of object i in a text j. Though df_{ji} specify the docs count that term i has checked, and n is the total number of documents present in a corpus. The statistical measurement of the word weight under this method tests the frequency of presence in a text for one word in addition to the entire corpus. Whether a phrase exists, it is most frequently mentioned as a stop-word in various documents, TFIDF eliminates those words when the TFIDF score changes to zero or near zero for stop-words [38, 39].'

Experiment and Results for Good Number of k Determination

The data used for our experiment is retrieved from the whole requirements dataset from Functional requirements document for ERP ecosystem [40], and the researchers tried both Average within cluster distance and Davies-Bouldin index to determine the most suitable number of K Clusters according to the below model design in Fig. 2 and 3.

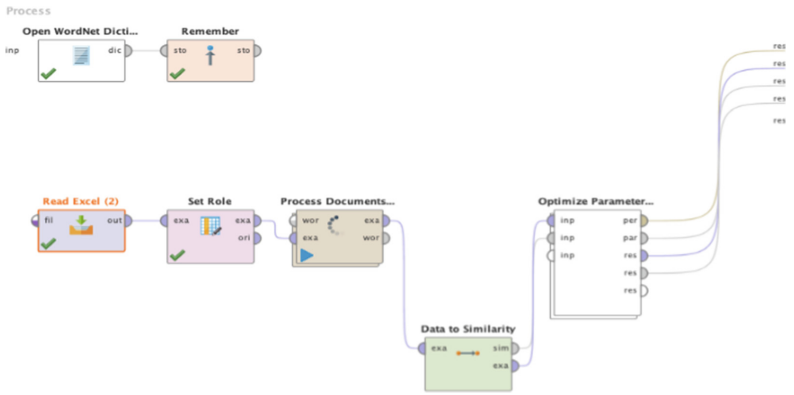

Fig. 2. High Level Model Design for Finding Most Suitable "K Cluster Value"

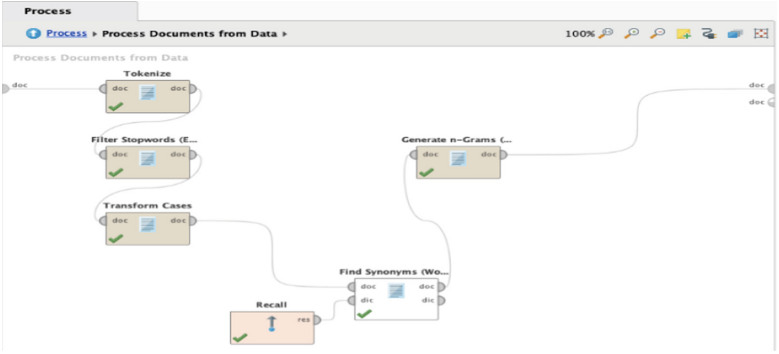

Fig. 3. Detailed Tasks View for Finding the Most Suitable 'K Cluster Value'

We have tested the proposed two methods, the first method – Average Centroid Distance as Silhouette Index – is the one shown in Figs. 1, 2, 3, and 4 showing the optimum number of k is 70 Clusters with average requirements similarity 0.586 and 250 attributes, and the average similarity is described in detail for each cluster in Fig. 5 after many iterations in order find the good number of k extracted from model output through RapidMiner [41].

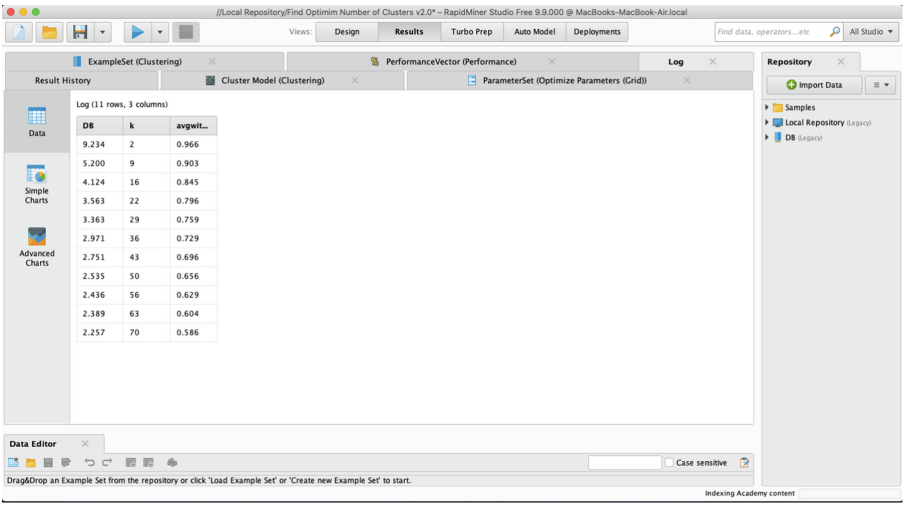

Fig. 4. Different k clusters with their average similarity using Average Centroid Distance

```
Parameter set:

Performance:
PerformanceVector [
*****Avg. within centroid distance: -0.586
-----Avg. within centroid distance_cluster_0: 0.712
-----Avg. within centroid distance_cluster_1: 0.668
-----Avg. within centroid distance_cluster_2: 0.547
-----Avg. within centroid distance_cluster_3: 0.635
-----Avg. within centroid distance_cluster_4: 0.712
-----Avg. within centroid distance_cluster_5: 0.372
-----Avg. within centroid distance_cluster_6: 0.749
-----Avg. within centroid distance_cluster_7: 0.765
-----Avg. within centroid distance_cluster_8: 0.626
-----Avg. within centroid distance_cluster_9: 0.613
-----Avg. within centroid distance_cluster_10: 0.764
-----Avg. within centroid distance_cluster_11: 0.672
-----Avg. within centroid distance_cluster_12: 0.664
-----Avg. within centroid distance_cluster_13: 0.840
-----Avg. within centroid distance_cluster_14: 0.627
-----Avg. within centroid distance_cluster_15: 0.749
-----Avg. within centroid distance_cluster_16: 0.677
-----Avg. within centroid distance_cluster_17: 0.228
-----Avg. within centroid distance_cluster_18: 0.765
-----Avg. within centroid distance_cluster_19: 0.395
-----Avg. within centroid distance_cluster_20: 0.656
-----Avg. within centroid distance_cluster_21: 0.627
-----Avg. within centroid distance_cluster_22: 0.725
-----Avg. within centroid distance_cluster_23: 0.411
-----Avg. within centroid distance_cluster_24: 0.767
-----Avg. within centroid distance_cluster_25: 0.308
-----Avg. within centroid distance_cluster_26: 0.355
-----Avg. within centroid distance_cluster_27: 0.422
-----Avg. within centroid distance_cluster_28: 0.660
-----Avg. within centroid distance_cluster_29: 0.000
-----Avg. within centroid distance_cluster_30: 0.268
-----Avg. within centroid distance_cluster_31: 0.463
-----Avg. within centroid distance_cluster_32: 0.359
-----Avg. within centroid distance_cluster_33: 0.555
-----Avg. within centroid distance_cluster_34: 0.470
-----Avg. within centroid distance_cluster_35: 0.000
-----Avg. within centroid distance_cluster_36: 0.584
-----Avg. within centroid distance_cluster_37: 0.742
-----Avg. within centroid distance_cluster_38: 0.468
-----Avg. within centroid distance_cluster_39: 0.501
-----Avg. within centroid distance_cluster_40: 0.753
-----Avg. within centroid distance_cluster_41: 0.556
-----Avg. within centroid distance_cluster_42: 0.294
-----Avg. within centroid distance_cluster_43: 0.399
-----Avg. within centroid distance_cluster_44: 0.249
-----Avg. within centroid distance_cluster_45: 0.520
-----Avg. within centroid distance_cluster_46: 0.686
-----Avg. within centroid distance_cluster_47: 0.660
-----Avg. within centroid distance_cluster_48: 0.130
-----Avg. within centroid distance_cluster_49: 0.531
-----Avg. within centroid distance_cluster_50: 0.645
-----Avg. within centroid distance_cluster_51: 0.350
-----Avg. within centroid distance_cluster_52: 0.656
-----Avg. within centroid distance_cluster_53: 0.636
-----Avg. within centroid distance_cluster_54: 0.725
-----Avg. within centroid distance_cluster_55: 0.701
-----Avg. within centroid distance_cluster_56: 0.638
-----Avg. within centroid distance_cluster_57: 0.434
-----Avg. within centroid distance_cluster_58: 0.477
-----Avg. within centroid distance_cluster_59: 0.488
-----Avg. within centroid distance_cluster_60: 0.666
-----Avg. within centroid distance_cluster_61: 0.255
-----Avg. within centroid distance_cluster_62: 0.286
-----Avg. within centroid distance_cluster_63: 0.447
-----Avg. within centroid distance_cluster_64: 0.697
-----Avg. within centroid distance_cluster_65: 0.689
-----Avg. within centroid distance_cluster_66: 0.324
-----Avg. within centroid distance_cluster_67: 0.421
-----Avg. within centroid distance_cluster_68: 0.296
-----Avg. within centroid distance_cluster_69: 0.584
]
Clustering.k          = 70
```

Fig. 5. Average Similarity Centroid per each Cluster

The second method – Davies-Bouldin Index evaluates intra-cluster similarity and inter-cluster differences – is the one shown in Figs. 1, 2, 3, 4 and 5 showing the optimum number of k is 70 Clusters with average requirements similarity 0.586, and the average similarity is described in detail for each cluster in Fig. 7.

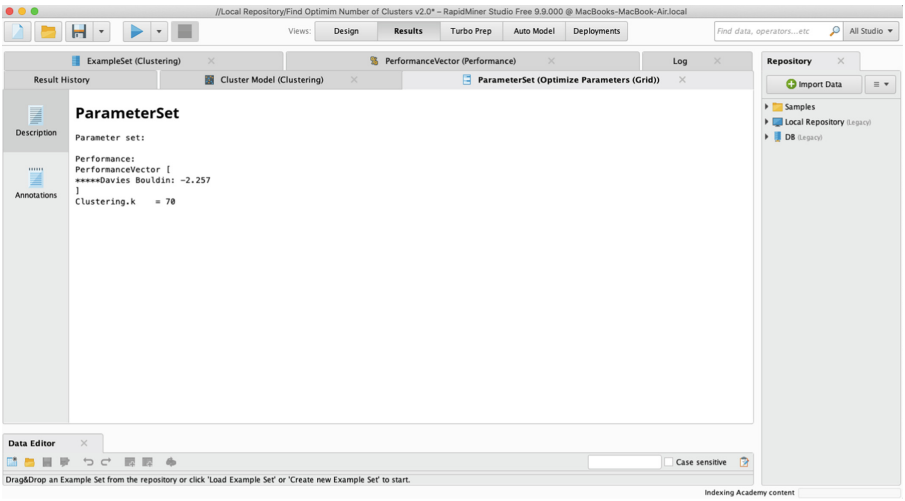

Fig. 6. Recommended Number of K by Davies Bouldin

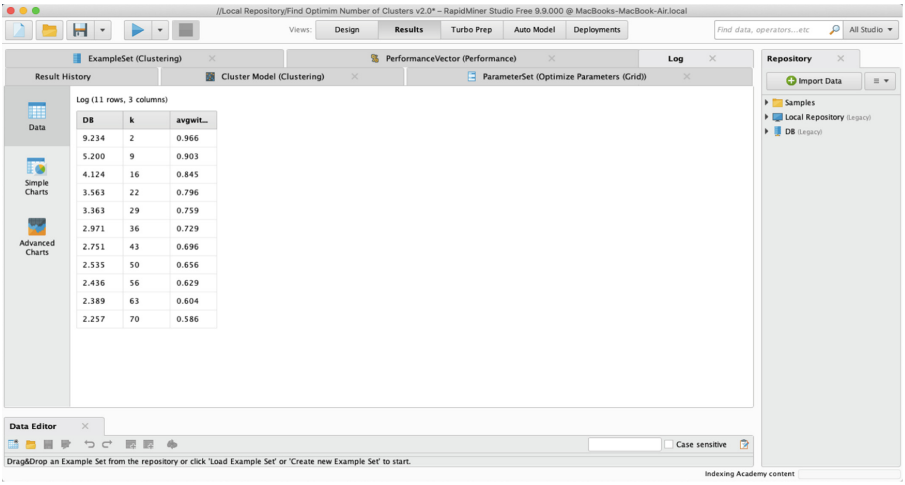

Fig. 7. Different k clusters with their average similarity using Davies Bouldin

5 K Clustering Model Implementation

The following chart illustrates a High-level functional model for how to create the requirements clusters using the calculated similarity score among Systems Requirement R after passing through different activities, then retrieve all the dependent set of requirements those could be affected by any in-place requirements amendment or deletion.

In the model, finding the interconnected set of requirements as end results are passed through two main steps, first step (Requirements Clustering Generation) all requirement

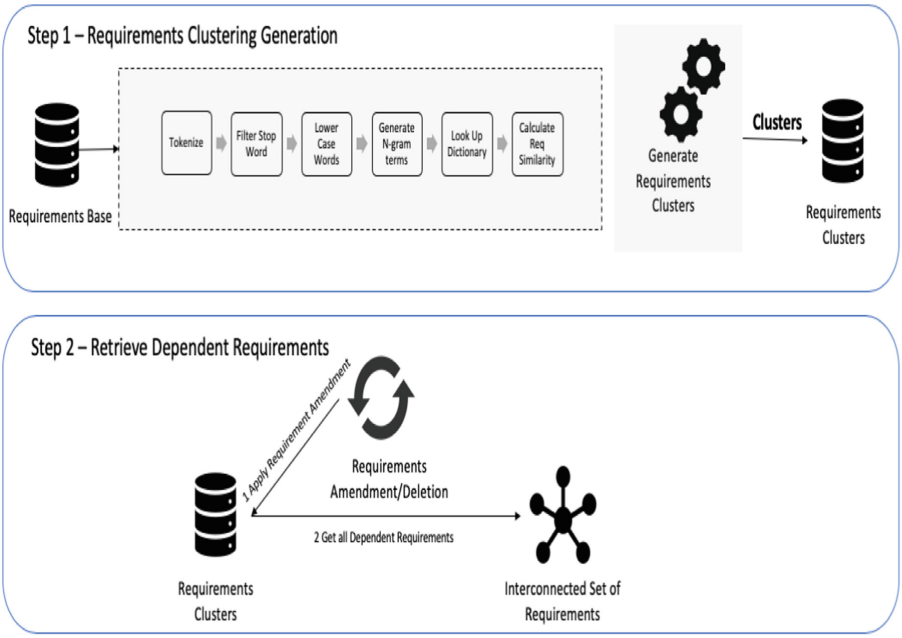

Fig. 8. Model Implementation

documents which are placed in the Requirements base, are passing through different document processing activates which were described, in order to calculate the similarity score among all requirements considering the semantic meaning for requirements vectors. Afterwards, all the textual requirements are allocated or labeled to a cluster number according to the automatic generated number of clusters passed to the model from previous section. The purpose of this step is to allocate each requirement to a k cluster number among similar requirements. Finally, in the second step (Get all Dependent Requirements), once the stakeholders decide to amend or delete any single or group of requirements, the list of interconnected or dependent requirements associated in same cluster(s) are presented as proper affected requirements.

5.1 Detailed Model Data Examination

Aiming to investigate the potential benefits of the clustering model, we have applied the k clustering on the real industrial requirements for around 4000 Requirements Statements.

The following section describes each process in detail among the flow of activities applied in RapidMiner [41] for the requirements repository in order to allocate them to different clusters.

Remember Wordnet Dictionary

This operator utilized to save the wordnet dictionary into the object store of the activity in order to minimize the I/O process for each vector during the model run. The saved

object would be restored in the future using Recall function mentioning the same name and class.

Read Data
This function used to read data from Microsoft Excel sheets, where the whole requirements are stored in spreadsheets. The table must be arranged in such a way that each row is an example, and each column is an attribute. For attribute titles which could be specified as parameters, see the first row of the Excel sheet.

Tokenize
Mainly, that operator separates the document's text to sequence of tokens. There are numerous choices just how to determine the segregation themes. Whichever it can be used all non-letter character. The mode that was used is to separate the text into English linguistic tokens.

Filters Stop Words (English)
This function filters out English stop words from a document by eliminating every token that matches a stop word from the embedded stop word list. Every token should represent a single English word only.

Transform Case
This operator transforms all characters in a document to either lower case or upper case, correspondingly.

Recall Dictionary
This operator recalls the itemized wordnet dictionary from the object recall of the procedure. The objects can be held in the object store by using the Remember Operator.

Find Synonyms
This operator uses wordnet dictionary file, as it received the text vector and find all the synonym or different meaning for each word or vector.

Generate n-gram
This function produces tokens in a document with term n-Grams. It defines a word n-Gram as a series of repeated tokens of length n. The term n-Grams generated by this operator includes all the following series of n-length tokens, the default length being 2.

Data to Similarity
The "Data to Similarity Data" operator measures the similarity of an Example Set among all requirements. Even requirements are comparable to those of themselves. So, if the Example Set includes n examples of specifications, this operator returns similarity comparisons to n^2. The Data to Similarity Data Operator returns an Example Set called a view, so that there will be no memory errors here.

K-means Clustering
This Operator performs the clustering using k-means algorithm. Clustering groups Examples together which are similar to each other, Clustering can be used on un-labelled data and is an algorithm of unsupervised machine learning.

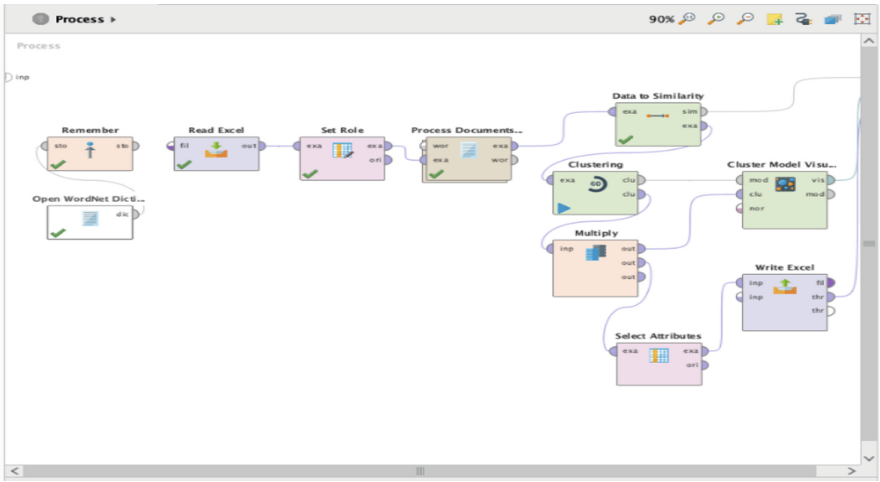

Fig. 9. Clustering Model Technical Description

6 Clustering Model Results

As explained, k-means clustering is used to see how the model can correctly detect if a certain requirement itself has changed or deleted during ULS system run and get the auto discovered or dependent set of requirements those could be impacted.

So, the following Figs. 11, 12 and 13 show the allocation of requirements into several clusters, each cluster contains at least one to n requirements.

Cluster Model

Cluster 0: 78 items
Cluster 1: 50 items
Cluster 2: 29 items
Cluster 3: 26 items
Cluster 4: 30 items
Cluster 5: 36 items
Cluster 6: 28 items
Cluster 7: 96 items
Cluster 8: 52 items
Cluster 9: 30 items
Cluster 10: 82 items
Cluster 11: 29 items
Cluster 12: 22 items
Cluster 13: 23 items
Cluster 14: 25 items
Cluster 15: 57 items
Cluster 16: 29 items
Cluster 17: 35 items
Cluster 18: 77 items
Cluster 19: 40 items
Cluster 20: 47 items
Cluster 21: 44 items
Cluster 22: 37 items
Cluster 23: 42 items
Cluster 24: 32 items
Cluster 25: 35 items
Cluster 26: 28 items
Cluster 27: 19 items
Cluster 28: 26 items
Cluster 29: 48 items
Cluster 30: 37 items
Cluster 31: 27 items
Cluster 32: 31 items
Cluster 33: 34 items
Cluster 34: 53 items
Cluster 35: 26 items
Cluster 36: 32 items
Cluster 37: 47 items
Cluster 38: 35 items
Cluster 39: 137 items
Cluster 40: 39 items
Cluster 41: 46 items
Cluster 42: 38 items
Cluster 43: 16 items
Cluster 44: 68 items
Cluster 45: 20 items
Cluster 46: 45 items
Cluster 47: 47 items
Cluster 48: 41 items
Cluster 49: 37 items
Cluster 50: 16 items
Cluster 51: 53 items
Cluster 52: 28 items
Cluster 53: 25 items
Cluster 54: 25 items
Cluster 55: 42 items
Cluster 56: 27 items
Cluster 57: 89 items
Cluster 58: 42 items
Cluster 59: 14 items
Cluster 60: 34 items
Cluster 61: 50 items
Cluster 62: 1 items
Cluster 63: 41 items
Cluster 64: 53 items
Cluster 65: 69 items
Cluster 66: 25 items
Cluster 67: 62 items
Cluster 68: 66 items
Cluster 69: 48 items
Total number of items: 2898

Fig. 10. Requirements allocation to Clusters

The below Fig. 13 shows if there is any change on a specific requirement, all the cluster related requirements have the same probability of propagation since they belong to same cluster.

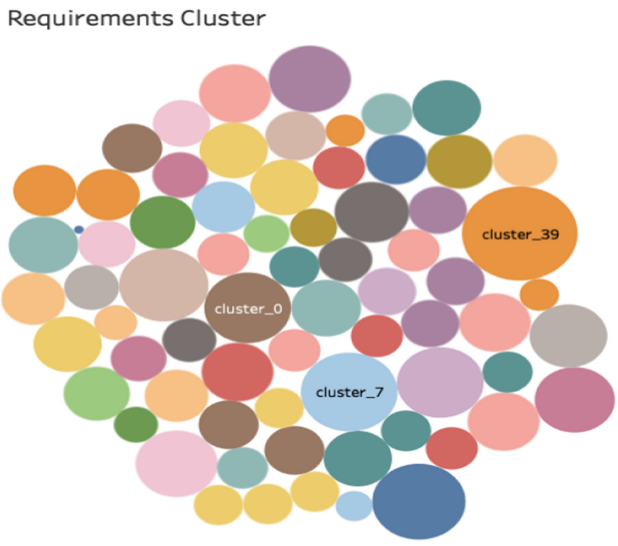

Fig. 11. Clusters Size

Cluster Details

Req Seq	Label	Req Text
R0001	cluster_48	All systems should be integrated with other subsyste
R0002	cluster_48	Provide integration with other critical systems to pro
R0003	cluster_58	All systems must be able to produce an electronic file
R0004	cluster_37	Security is required for each application with the abili
R0005	cluster_49	Limit based on role and allow exceptions using role-ba
R0006	cluster_7	Provides application and system performance measur
R0007	cluster_67	Ability to generate report files in delimited, ASCII, PD
R0008	cluster_46	Provides field level edits to ensure validity of the data
R0009	cluster_37	Help is searchable, editable and County specific.
R0010	cluster_33	Ability to request reports on-line for immediate or def
R0011	cluster_48	On-line training and demo module included with appli
R0012	cluster_67	Ad-hoc report writer packaged with application softw
R0013	cluster_67	Provide GUI-based end-user report viewing and query
R0014	cluster_7	System supports electronic workflow throughout all s
R0015	cluster_48	Integrated systems pass transactions, data and infor
R0016	cluster_32	Ability to view audit trail via screens and reports. Use
R0017	cluster_32	Ability to archive audit trail data
R0018	cluster_32	Ability to record audit trail for all data updates and sp
R0019	cluster_45	Ability to link to files located in a document managem
R0020	cluster_58	Supports electronic signatures
R0021	cluster_45	Document and diagrammatically represent workflows
R0022	cluster_39	Permits easy modification of work flows by end user c
R0023	cluster_49	Supports role-based workflow levels (different user, s
R0024	cluster_39	Changes to the work flow approval path do not affect
R0025	cluster_38	Workflow items that are in-process are able to be inqu
R0026	cluster_39	End users have an individual workflow-driven work qu
R0027	cluster_37	Automated e-mail notification for specific events and,
R0028	cluster_38	Inquire on open approval items in a workflow approva
R0029	cluster_55	Supports single sign-on
R0030	cluster_3	Supports different security structure for internal cou
R0031	cluster_63	Limit enforcement of transaction execution via a com
R0032	cluster_56	Limit enforcement of transaction execution via a com
R0033	cluster_7	System Administrator can view list of logged-in users
R0034	cluster_49	Supports group level permissions
R0035	cluster_3	Produces user security profile report Limit access by s

Fig. 12. Example of Cluster 0 and its allocated requirements IDs

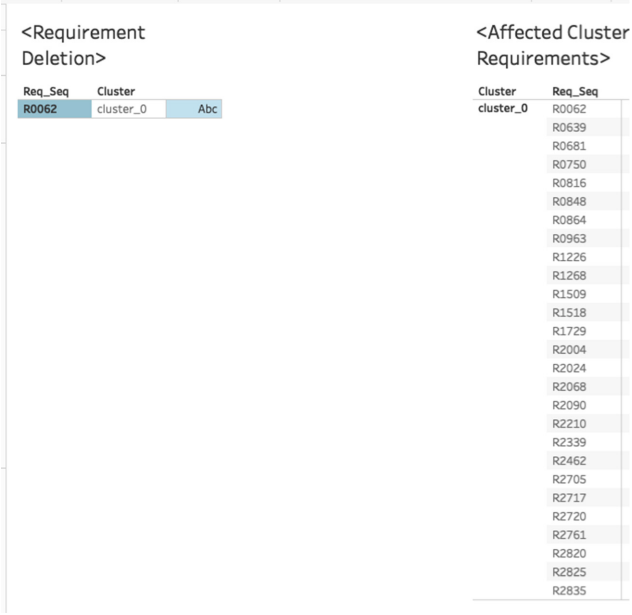

Fig. 13. Change in one single requirement and its impact on the whole cluster's requirements

7 Conclusions

In this work, we proposed a model to measure the impact of change request submitted in a free natural language against a large set of requirements. The key characteristics of the approach are that it exploits the synonyms of text in which the user is representing the change against the list of requirements through NLP similarity models.

Automated similarity analysis is used widely in many different business areas started by search engines, recommender systems and sentiment analysis and become a promising technique for supporting requirements engineers to manage requirements evolution since it has been managed in a decentralized way by crowdsourcing stakeholders and development team in ecosystems.

The model showed noticed results since there are no control over neither the structure of written textual requirement, the change requests, the formal semantics of requirements relation types or even the domain knowledge of the system. In addition, it can run over bulk of changes simultaneously with no barrier statements volume.

Since, it's an unsupervised prediction model, consequently, there is no predefined training data set to feed the model, so it's scaling up wherever needed.

We have implemented our model on RapidMiner, it's a data science application for machine learning, data mining, text mining, predictive analytics, and business analytics [41]. Each function in the suggested model is represented in one more task in the tool. Figures illustrate the tasks flow starting from reading each of requirements and changes repositories, applying the similarity model till providing the results at the end through a table showing the distance value between each change and requirement.

Finally, in this paper, we described a feasible model that utilizes unsupervised cluster technique and similarity models to scale-up the change impact analysis model of ULS requirements change management. We believe if combined both models change impact analysis problems can be solved in a noticeable way, we would be in a much stronger situation to challenge some of the higher-level difficulties identified in the ULS Systems report [42], like those related to unbalanced requirements, immediate requirements, and continues changes that happen alongside different interconnected system.

References

1. Functional Requirements for Enterprise Managment Systems. https://mulco.us/file/47449/download
2. Felfernig, A., Ninaus, G., Reinfrank, F., Weninger, L., Pagano, D., Maalej, W.: An Overview of Recommender Systems in Requirements Engineering, in Managing Requirements Knowledge, pp. 316–318. Springer, Munich (2013)
3. Burke, R.: Hybrid recommender systems: survey and experiments. UMUAI **12**(4), 331–370 (2002)
4. Terveen, L., Herlocker, J., Konstan, J., Riedl, H.: Evaluating collaborative filtering recommender systems. ACM Trans. Inf. Syst. **22**(1), 5–53 (2004)
5. Linden, G., Smith, B., York, J.: Amazon.com recommendations: item-to-item collaborative filtering. IEEE Inter. Comput. **7**(1), 76–80 (2004)
6. Pazzani, M., Billsus, D.: Learning and revising user profiles: the identification of interesting web sites. Mach. Learn. **27**, 313–331 (1997)
7. Felfernig, A., Burke, R.: Constraint-based recommender systems: technologies and research issues. In: Proceedings of IEEE ICEC'08, Innsbruck (2008)
8. Zayed, R.A., Ibrahim, L.F., Hefny, H.A., Salman, H.A.: Shilling attacks detection in collaborative recommender system: challenges and promise. In: Barolli, L., Amato, F., Moscato, F., Enokido, T., Takizawa, M. (eds.) WAINA 2020. AISC, vol. 1150, pp. 429–439. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-44038-1_39
9. Soo, L., Cornelius, N.: Social networks and crowdsourcing for stakeholder analysis in system of systems projects. In: 8th International Conference on System of Systems Engineering, Hawaii (2013)
10. Mulla, N.: A new approach to requirement elicitation based on stakeholder recommendation and collaborative filtering. Int. J. Softw. Eng. Appl. **3**(3), 51–60 (2012). https://doi.org/10.5121/ijsea.2012.3305
11. Dumitru, H., Gibiec, M., Hariri, N., Cleland-Hunang, J. Castro-Herrera, C.: On-demand feature recommendations derived from mining public product descriptions. In: Proceedings of ACM/IEEE, Waikiki/Honolulu (2011)
12. Lim, S.L., Finkelstein, A.: StakeRare: using social networks and collaborative filtering for large-scale requirements elicitation. IEEE Trans. Softw. Eng. **38**(3), 707–735 (2012). https://doi.org/10.1109/TSE.2011.36
13. Mobasher, B., Cleland-Huang, J.: Recommender systems in requirements engineering. AI Mag. **32**(3), 81–89 (2011). https://doi.org/10.1609/aimag.v32i3.2366
14. Soo, L., Daniela, D., Anthony, F.: StakeSource2.0: using social networks of stakeholders to identify and prioritise requirements. In: ICSE'11 Proceedings of the 33rd International Conference on Software Engineering, pp. 1022–1024 (2011)
15. Chuan, D., Paula, L., Cleland-Huang, J., Kwiatkowski, C.: Towards automated requirements prioritization and triage. Requirements Eng. **14**(07), 73–89 (2009)

16. Felfernig, A., Schubert, M., Mand, M., Ghirardini, P.: Diagnosing inconsistent requirements preferences in distributed software projects. In: Proceedings of 3rd International Workshop on Social Software Engineering, Paderborn (2009)
17. Cleland-Huang, J., Dumitru, H., Duan, C., Castro-Herrera, C.: Automated support for managing feature requests in open forums. Commun. ACM **52**(10), 68–74 (2009)
18. Rick, S., Marsha, C., Jennifer, H., Alessio, D.S.: Managing requirements uncertainty with partial models. Requirements Eng. **18**(2), 105–106 (2013)
19. Castro-Herrera, C., Duan, C., Cleland-Huang, J., Mobasher, B.: Using data mining and recommender systems to facilitate large-scale, open, and inclusive requirements elicitation processes. In: Proceeding of the 16th IEEE international conference on requirements engineering (RE'08), Barcelona (2008)
20. Fantechi, A., Spinicci, E.: A content analysis technique for inconsistency detection in software requirements documents. In: WER05 – workshop em Engenharia de Requisitos, Porto (2005)
21. Belsis, P., Koutoumanos, A., Sgouropoulou, C.: PBURC: A patterns-based, unsupervised requirements clustering framework for distributed agile software development. Requirements Eng. **19**(2), 213–225 (2013). https://doi.org/10.1007/s00766-013-0172-9
22. Anna, H.: Similarity Measures for Text Document Clustering (2008)
23. Salton, G.: Automatic Text Processing. Addison-Wesley (1989)
24. Allahyari, M., et al.: A brief survey of text mining: classification, clustering and extraction techniques. In: KDD Bigdas, Halifax, Canada, August 2017
25. Ushioda, A.: Hierarchical clustering of words and application to NLP Tasks. In: Fourth Workshop on Very Large Corpora, Association for Computational Linguistics, pp. 28–41 (1996)
26. Can, F., Ozkarahan, A.: Concepts and effectiveness of the clustering methodology for text databases. ACM Trans. Database Syst. **15**(4), 483–517 (1990)
27. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: a review. ACM Comput. Surv. **31**(3), 264–323 (1999). https://doi.org/10.1145/331499.331504
28. Duan, C.: Clustering and its Application in Requirements Engineering. College of Computing and Digital Media, Via Sapientiae (2008)
29. Smyth, P.: Clustering using monte carlo cross validation. In: 2nd International Conference Knowledge Discovery and Data Mining (KDD-96), Portland (1996)
30. Naeem, S., Wumaier, A.: Study and implementing K-mean clustering algorithm on english text and techniques to find the optimal value of K. Int. J. Comput. Appl. **182**(31), 7–14 (2018)
31. Davies, D.L., Bouldin, D.W.: A cluster separation measure. IEEE Trans. Pattern Anal. Mach. Intell. **PAMI-1**(2), 224–227 (1979). https://doi.org/10.1109/TPAMI.1979.4766909
32. Siddheswar, R., Rose, H.: Determination of Number of Clusters in K-Means Clustering and Application in Color Image Segmentation (1998)
33. Julie, B.: Development of a stemming algorithm. MIT Information Processing Group, Electronic Systems Laboratory (1968)
34. Christopher, D., Prabhakar, R., Hinrich, S.: An Introduction to Information Retreival, p. 26. England, Cambridge University Press, Cambridge (2007)
35. Core, R.: Cluster Distance Performance (2018). https://docs.rapidminer.com/8.0/studio/operators/validation/performance/segmentation/cluster_distance_performance.html. Accessed 11 2019
36. Rapidminer: Cluster Distance Performance. Rapidminer, Jan 2020
37. Li, S., Liu, C., Wang, Y. (eds.): CCPR 2014. CCIS, vol. 484. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45643-9
38. Mehdi, A.: A Brief Survey of Text Mining: Classification Clustering and Extraction Techniques in KDD Bigdas. Halifax, Canada (2017)
39. Shah, N., Mahajan, S.: Document clustering: a detailed review. Int. J. Appl. Inform. Syst. **4**(5), 30–38 (2012)

40. Functional Requirements for ERP Ecosystem (Cuyahoga County Government Functional Requirements). http://it.cuyahogacountry.us. Accessed Oct 2019
41. R. Miner. www.rapidminer.com
42. Feier, P., et al.: Ultra Large Scale Systems: The Software Challenge of the futhure. Technical Report, Software Engineering Institue (2006)