A Grouping Strategy based Partition Algorithm for Clustering Questions in a Question Bank

[1]Dimple V. Paul, [2]Jyoti D. Pawar
[1, First Author] *Department of Computer Science, DM's College of Arts, Science and Commerce, Asssagao, Bardez, Goa,403 507. dimplevp@rediffmail.com*
[*2,Corresponding Author] *Department of Computer Science and Technology, Goa University, Taleigao Plateau, Goa, India, 403206. jyotidpawar@gmail.com*

Abstract

Grouping algorithms are widely used in multidisciplinary fields such as data mining, image analysis and bioinformatics. This paper proposes the use of Grouping Strategy based Partition Algorithm for clustering the questions in a Question Bank. It includes a new approach for computing the question similarity matrix and use of the matrix in clustering the questions. The grouping algorithm extracts n module-wise questions, compute n × n similarity matrix by performing n × (n-1)/2 pair-wise question vector comparisons and uses the matrix in formulating question clusters. Grouping algorithm has been found efficient in reducing the best case time complexity, O (n× (n-1)/2 log n) of hierarchical approach to O (n × (n-1)/2). Experimental study was carried out by performing a comparative analysis of question clusters formulated with two different similarity measures. Performance evaluation using F-measure proves that grouping strategy based partition algorithm is efficient in formulating question clusters without the initial specification of the number of clusters as well as the iterative stages of cluster formulation.

Keywords: *Cosine Similarity, Jaccard Coefficient, Similarity Matrix, Question Bank, Grouping Strategy, Partition Algorithm*

1. Introduction

Subject wise Question Banks (QB) serves as organized repositories for storing objective and descriptive questions under different modules/ topics. The set of questions within a module may or may not have similarity among its contents. During automatic question paper generation, it is necessary to detect the percentage of similarity among questions and thereby avoid question conflicts. In order to successfully carry out clustering of similar questions, we propose a similarity matrix based grouping algorithm. Similarity matrix is generated by computing the similarity of each pair of questions based on different similarity measures. There are several similarity measures for finding groups of related questions in a given QB. The proposed grouping strategy based partition algorithm is an alternative to the most widely used Agglomerative Hierarchical Clustering approach. The Agglomerative Hierarchical Clustering starts with a similarity matrix and generates clusters by iteratively merging the two clusters that are most similar at each of the iteration. The iterative stages are continued until the desired numbers of clusters are found or only one cluster remains. Our grouping strategy based partition algorithm also starts with a similarity matrix but avoids the iterative stages of hierarchical clustering. It is achieved by formulating the disjoint clusters of similar questions in parallel with the generation of each row of the similarity matrix. Hence, the best case time complexity, O *(n× (n-1)/2 log n)* of hierarchical clustering can be reduced to *O (n× (n-1)/2)* by introducing grouping algorithm. This paper is organized as follows. Section 2 introduces the research background and related work, section 3 explains the methodology used for similarity matrix representation, section 4 explains the problem formulation, section 5 highlights the implementation details and sections 6 presents conclusion and future work.

2. Research background & Related Work

Cluster analysis aims to organize a collection of patterns into clusters based on their similarity. Document clustering is particularly useful in many applications such as automatic categorization of documents, grouping search engine results, building taxonomy of documents etc[1][2]. Document

clustering algorithms generally use four major steps such as pattern recognition, pattern proximity determination, grouping and output evaluation. Pattern recognition identifies the number of patterns and features contained in the query specification. Pattern proximity determines the best similarity formula to be used in the clustering algorithm. Grouping is an iterative process which applies the similarity formula, computes the pair-wise similarity of each document with the rest of the documents and generates new clusters of documents based on the specified threshold value. Output evaluation is generally carried out using F-measure. F-measure combines two different retrieval measures such as precision and recall into one measure. Precision is defined as the measure of relevant versus non-relevant items returned. In terms of document clustering, precision is the measure of high similarity of the documents in the cluster versus low similarity with documents in other clusters. Recall is defined as the measure of relevant documents retrieved versus relevant documents not retrieved. In terms of document clustering, recall is used to measure the lack of documents in other clusters whose individual similarity to documents in another cluster is high [3].

Document clustering techniques mostly are based on single term analysis of the document data set, such as the vector space model. The Vector Space Model (VSM) is a popular information retrieval system implementation which facilitates the representation of a set of documents as vectors in the term space [4]. Similarity matrix of a set of questions in a QB is a two dimensional matrix representing the pair-wise similarity of topic-wise questions. Pair-wise similarity computation can be performed on different similarity measures. We have used Cosine Similarity and Jaccard Similarity Coefficient to assign a similarity score to each pair of compared questions. The cosine function uses angular measure while jaccard uses association coefficient. Cosine similarity is a measure of similarity between two vectors by measuring the cosine of the angle between them. Cosine of the angle is generally 1.0 for identical vectors and is in the range of 0.0 to 1.0 for non-similar or partially similar vectors. Cosine similarity remains as the most popular measure because of its simple interpretation, easy computation and document length exclusion [5]. The performance efficiency of cosine in detecting the similarity of questions in a QB has been confirmed by tallying its results with Jaccard Coefficient. Jaccard coefficient is a statistical measure for comparing the similarity and diversity of documents. The Jaccard similarity can be used, when interested in binary differences between two or more documents. It is popularly used in research investigations which focus on the presence/absence between several objects. The Jaccard similarity coefficient ranges between 0 and 1. It is 1 when two question vectors are identical and zero when they are disjoint [6]. The grouping algorithm for clustering is a kind of partitioning algorithm [7]. It considers words of question vectors, applies cosine and jaccard similarity measures and computes question-similarity-matrix. During the process of computation of each row of the question-similarity-matrix, the grouping algorithm does parallel generation of question clusters by selecting the set of questions satisfying the input similarity threshold value. Hence, it does not require the initial specification of the number of clusters as well as the iterative stages of cluster formulation.

3. Methodology Used

Table 1. Terminology used

Term	Meaning
Question Bank (QB)	QB is a database which stores topic wise questions with its details such as question-no, question-content, question-type, question-marks and question-answer-time
N	N is the total number of questions stored under a topic
n_i	n_i refers to the number of questions in which term i appears
$freq_{ij}$	$freq_{ij}$ is the frequency of term i in question j
maximum frequency($maxfreq_{ij}$)	$maxfreq_{ij}$ is the maximum frequency of a term in question j
term frequency(tf_{ij})	tf_{ij} refers to the importance of a term i in question j. It is calculated using the formula- $tf_{ij} = \frac{freq_{ij}}{maxfreq_{ij}}$
Inverse Document Frequency(idf_i)	idf_i refers to the discriminating power of term i and is calculated as - $idf_i = \log_2 (N/n_i)$
tf-idf weighting(W_{ij})	It is a weighting scheme to determine weight of a term in a question. It is calculated using the formula- $W_{ij} = tf_{ij} \times idf_i$
Term-set(question q_i)	It is a set of terms extracted from each question by performing its tokenization, stop word removal, taxonomy verb removal and stemming

The procedure for finding similarity between module-wise/ topic-wise questions in a QB follows the steps as below-

1. Question Bank Pre-processing
2. Computing Question Similarity Matrix
3. Generating Question Clusters
4. Evaluating Question Clusters

A brief description of the approaches used for performing each of the above steps is as follows-

3.1. Question Bank Pre-processing

We handle the pre-processing of questions using the following four sub-steps-

3.1.1. Tokenization: Each question is treated as a collection of strings (or bag of words), which are then partitioned into a list of terms.

3.1.2. Filtering Stop Words: Stop words are frequently occurring, insignificant words within the question content and are eliminated.

3.1.3. Filtering Taxonomy Verbs: Educational researcher Benjamin Bloom and colleagues have suggested six different cognitive stages in learning such as Knowledge, Comprehension, Application, Analysis, Synthesis and Evaluation. Details of verbs and question examples that represent intellectual activity at each level of Bloom's taxonomy can be found in [9][10]. The taxonomy verbs within the question content are identified and eliminated.

3.1.4. Stemming Words: Stemming is a heuristic process of cutting off the ends of terms for getting the correct root form of the term. There are various word stemmers [11] available for English text and the most commonly used Porter stemmer is considered.

3.1.5. Normalization: The idea behind normalization is to convert all terms which mean the same, but written in different forms (e.g. CPU and C.P.U) into the same form. We are using the following techniques for performing normalization-

- Lowercase the terms
- Remove special characters

3.2. Computing Question Similarity Matrix

The similarity matrix computation is carried out by considering the matrix representation of vectors which is a natural extension of the existing VSM. Matrix representation of module-wise questions of a QB considers each question as a vector of terms. In the multidimensional matrix of N questions say N × N matrix, each pair of question gets compared to determine how identical they are by using different similarity measures. We have considered Cosine similarity and Jaccard similarity functions for similarity matrix computation. The cosine similarity measures the angle between two question-term vectors, but the jaccard coefficient measures similarity between two question-term sets. The term weight scores are calculated according to *tf-idf* weighting method [12][13]. *tf-idf* is the most commonly used scheme to assign weights to individual terms based on their importance in a collection of module-wise questions. Each weight score is calculated as a product of *tf* and *idf*. Higher values of *idf* correspond to terms which characterize a question more distinctly than others. Detail description of *tf-idf* calculation is shown in Table 1.

The cosine similarity of two question vectors say q_1 and q_2 is calculated by performing the dot product of each questions' term vectors. The calculation of cosine similarity is performed using the following formula-

$$Similarity\ (q_1, q_2) = cos\ \theta = \frac{q_1.q_2}{|q_1|.|q_2|} \quad (1)$$

, where '.' denotes the dot product between vectors q_1 and q_2. $|q_1|$ and $|q_2|$ are the Euclidean norm of q_1 *and* q_2 vectors. The above formula can be expanded in the following manner.

$$cos\ \theta = \frac{\sum_{i=1}^{n} Wt_{i,q1} \times Wt_{i,q2}}{sqrt(\sum_{i=1}^{n} Wt_i * Wt_{i,q1}) \times sqrt(\sum_{i=1}^{n} Wt_i * Wt_{i,q2})} \quad (2)$$

Jaccard similarity measure compares the sum weight of shared terms to the sum weight of terms that are present in either of the two questions but are not the shared terms. The jaccard similarity calculation is carried out using the following formula-

$$Similarity\ (q_1, q_2) = J(q_1, q_2) = \frac{q_1 \cap q_2}{(q_1 \cup q_2) - (q_1 \cap q_2)} \quad (3)$$

3.3. Generating Question Clusters

The grouping strategy based partition algorithm performs clustering of similar questions during the process of computation of pair-wise similarity of questions. Sample of a similarity matrix with the computed pair-wise similarity say $sm_{x,y}$ for n questions is represented in Table 2 below. The computation of similarity of n question is carried out by calculating similarity of $n \times (n-1)/2$ pairs of question vectors.

Table 2. Similarity Matrix Representation

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	…	Qn
Q1		sm_{12}	sm_{13}	sm_{14}	sm_{15}	sm_{16}	sm_{17}	…	sm_{1n}
Q2			sm_{23}	sm_{24}	sm_{25}	sm_{26}	sm_{27}		sm_{2n}
Q3				sm_{34}	sm_{35}	sm_{36}	sm_{37}		sm_{3n}
Q4					sm_{45}	sm_{46}	sm_{47}		sm_{4n}
Q5						sm_{56}	sm_{57}	…	sm_{5n}
Q6							sm_{67}	…	sm_{6n}
Q7									$sm_{7,n}$
…	…	…	…	…		…	…	…	…
Qn									

During the process of computation of the similarity matrix, its $sm_{x,y}$ value gets compared with the input threshold say ∂ in a row-wise manner. Result of the comparison is used as a standard for formulating similar question clusters. Hence, there is no additional time and space requirement for generating the clusters.

3.4. Evaluating Question Clusters

Precision, Recall and F-measure are commonly used as the metrics to evaluate the accuracy of predictions and the coverage of accurate pairs of comparisons while formulating the clusters. They are computed as –

$$\text{Precision (P)} = \frac{\text{Number of relevant question-to-question matches retrieved by the tool}}{\text{Total number of question-to-question matches retrieved by the tool}} \quad (4)$$

$$\text{Recall(R)} = \frac{\text{Number of relevant question-to-question matches retrieved by the tool}}{\text{Total number of question-to-question match given by the instructor}} \quad (5)$$

$$\text{F-measure} = \frac{(2 \times P \times R)}{P+R} \quad (6)$$

4. Problem Formulation

4.1. Problem Statement

Given a QB(S)={qb_1,qb_2,…,qb_N} where qb_i ={q_{i1},q_{i2},…,q_{ini}} is the question bank for module *i* of subject S consisting of n_i questions and N=the set of modules under subject S, the problem is to find clusters C_{i1},C_{i2},… of similar questions in qb_i.. A question q_{ij} is said to be similar to q_{ik} of module *i* if similarity ($q_{ij,}$ q_{ik})>= ∂ where ∂ is the user input threshold value to find the similarity. The similarity ($q_{ij,}$ q_{ik}) function could use any of the similarity measure available. We have used Cosine and Jaccard coefficient to perform the experimental study.

The main modules of this algorithm are shown below.

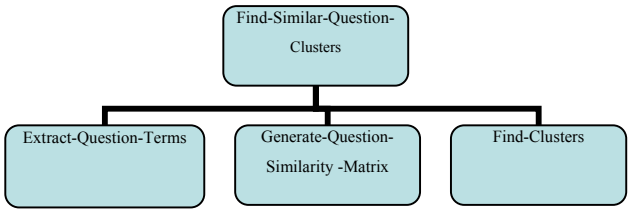


Figure 1. Main modules of question cluster formulation problem

The brief details of modules are presented below–

4.1.1 Extract-Question-Terms: Input q_{ij} (j=1 to n_i) and for each q_{ij} it extracts terms t_{ijs} (s=1 to n_{ij}).

4.1.2 Generate-Question-Similarity-Matrix: Input terms t_{ijk} (k=1 to n_{ij}) for all q_{ij} (j=1 to n_i). For each pair of questions q_{ij} and $q_{ik,}$ compute similarity ($q_{ij,}$ q_{ik}) for k=j+1 to n_i and represent it as Question-Similarity-Matrix.

4.1.3 Find-Clusters: Use Question-Similarity-Matrix and find clusters of similar questions satisfying similarity($q_{ij,}$ q_{ik})>= ∂.

4.2 Algorithm for Term Extraction

T_i= { }
For j=1 to n_i
{
Extract tokens from q_{ij} and store it in array t_{ij}[ ]
Remove the stop-words from t_{ij}[ ]
Remove taxonomy verbs from t_{ij}[ ]
Extract the stem of each token in t_{ij}[ ]
$T_{i=}$ T_i U t_{ij}[ ]
}
Output of Term Extraction, T_i= {t_{i1}[ ], t_{i2}[ ] t_{i3}[ ], …, t_{ini}[ ]}

4.3 Algorithm for Similar Question Cluster Formulation

Input: T_i= {t_{i1}[ ], t_{i2}[ ] t_{i3}[ ], …, t_{ini}[ ]} where
t_{ij} = { t_{ij1} , t_{ij2} , t_{ij3} ,…,t_{ijs}} for s=1 to n_{ij}
n_i ={q_{i1},q_{i2},…,q_{ini}}
Threshold ∂
Output: k clusters C_{i1},$C_{i2,…,}$ C_{ik} where C_{ij} consist of question
q'$_{ij}$ belongs to qb_i

```
Begin
  //Initialization
     p=0 // counter for new cluster formation
    cluster_set= [ ]
    //Similar-Question-Cluster-Formulation
   For j=1 to ni
    If qij not in cluster_set then
      p=p+1
      Cip=New-Cluster (qij, p)
      cluster_set= cluster_set + qij
      For k= (j+1) to ni
       //Similarity-Matrix-Computation
          If similarity (qij, qik) >=∂  then
                  Add qik to cluster k
            cluster_set= cluster_set + qik
          End If
      End For
    End If
  End For
End
```

5. Implementation Details

5.1. Hardware and Software Platform Used

Implementation is done using Microsoft Visual Basic .NET as Front End Tool and SQL Server as Back End Tool on a 2 GHz processor with 1GB RAM.

5.2. Datasets Used

Experimental data used is as follows-

5.2.1. S= Software Engineering (SE), a subject offered at the third year of the three year bachelor's degree course of computer science (B.Sc Computer Science) at Goa University.

5.2.2. QB(S) = $qb_{10,}$ where qb_{10} refers to module named Reengineering

5.2.3. qb10 ={q_{148},q_{149},…,q_{474}}

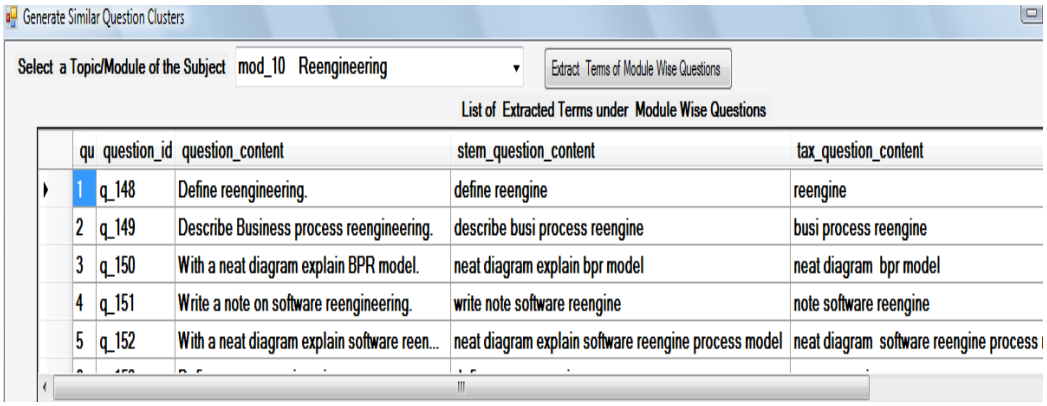
5.2.4. ∂=0.75

5.2.5. A snapshot of the set of questions {q_{148}, q_{149},…,q_{474}} with its extracted list of terms {reengine}, {busi, process, reengine}, {neat, diagram, bpr, model}, {note, software, reengine}, {neat, diagram, software, process, model} etc, for qb_{10} is displayed in Figure1. Extraction of the set of terms corresponding to qb_{10} is carried out by performing five different sub-steps of Question Bank Pre-processing such as Tokenization, Filtering Stop Words, Filtering Taxonomy Verbs, Stemming Words and Normalization.

5.3 Results Obtained

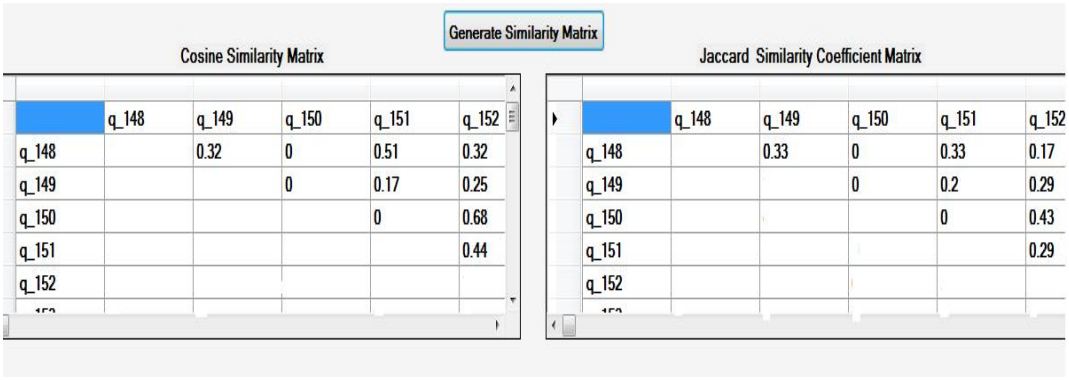
Figure 2 below shows sample screen shot of the *Question-Similarity-Matrix* computation for qb_{10} by using cosine similarity and jaccard similarity coefficient. Figure 3 represents the process of generation of question clusters. Each question q_{ij} of module qb_{10} in the *Question-Similarity-Matrix* sequentially gets compared with q_{ik} questions in the matrix based on the similarity measure, *similarity* ($q_{ij,}$ q_{ik}) >=∂.

If a question does not find its match with any of the generated question clusters, it formulates a *New-Cluster (),* identifies other similar questions and appends them to the new cluster. Figure 4 displays the comparative analysis of the question clusters formulated with cosine similarity and jaccard coefficient.



qu	question_id	question_content	stem_question_content	tax_question_content
1	q_148	Define reengineering.	define reengine	reengine
2	q_149	Describe Business process reengineering.	describe busi process reengine	busi process reengine
3	q_150	With a neat diagram explain BPR model.	neat diagram explain bpr model	neat diagram bpr model
4	q_151	Write a note on software reengineering.	write note software reengine	note software reengine
5	q_152	With a neat diagram explain software reen...	neat diagram explain software reengine process model	neat diagram software reengine process

Figure 1. The extracted list of terms under Re-engineering module of SE subject



	q_148	q_149	q_150	q_151	q_152
q_148		0.32	0	0.51	0.32
q_149			0	0.17	0.25
q_150				0	0.68
q_151					0.44
q_152					

	q_148	q_149	q_150	q_151	q_152
q_148		0.33	0	0.33	0.17
q_149			0	0.2	0.29
q_150				0	0.43
q_151					0.29
q_152					

Figure 2. Sample screen shot of the similarity matrix of Reengineering module using cosine similarity and jaccard similarity coefficient

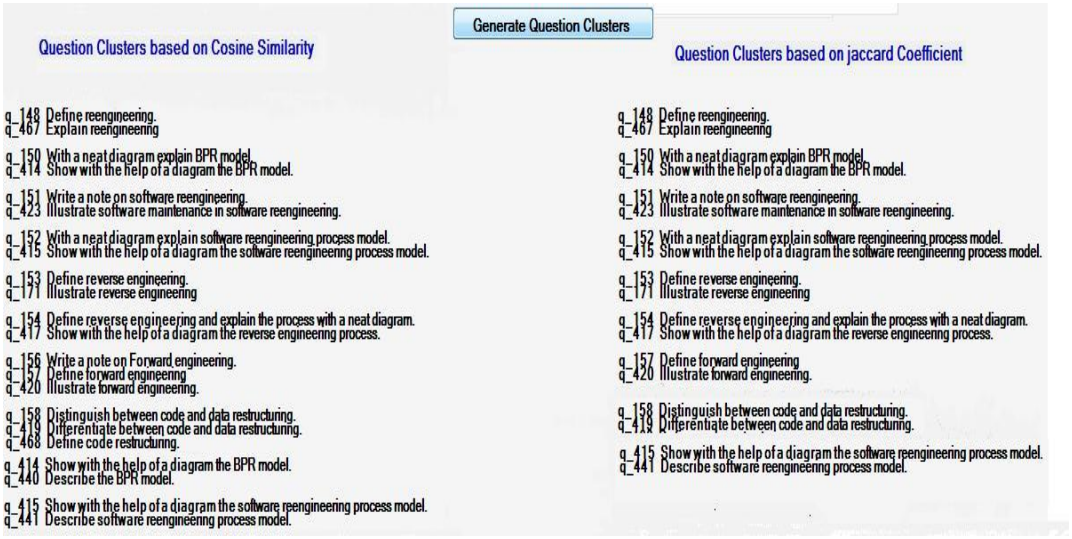


Figure 3. The extracted clusters of similar questions using Cosine Similarity and Jaccard Similarity Coefficient

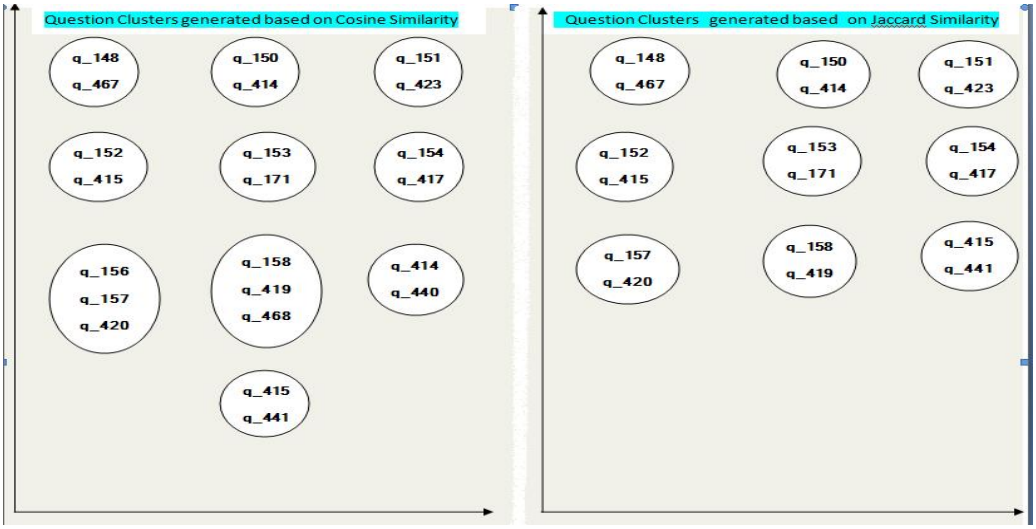


Figure 4. The extracted clusters of similar questions using Cosine Similarity and Jaccard Similarity Coefficient

5.4 Performance Evaluation

We have computed Precision, Recall and F-measure values in order to compare the performance of cosine similarity with jaccard similarity while formulating question clusters for the questions in QB of SE. Table 3 below shows the results of performance evaluation carried out by considering the question clusters formulated using four modules of SE such as reengineering, legacy systems, software testing and software configuration. Results indicate that cosine similarity outperforms jaccard in clustering similar questions.

Table 3. Performance Evaluation of SE Question Clusters

Module -of -the -subject	Cosine Precision	Cosine Recall	Cosine F-measure	Jaccard precision	Jaccard Recall	Jaccard F-measure
mod_10 reengineering	0.79	0.82	0.80	0.75	0.79	0.76
mod_11 legacy systems	0.78	0.81	0.79	0.73	0.78	0.74
mod_12 testing	0.77	0.82	0.79	0.72	0.77	0.74
mod_13 configuration	0.76	0.82	0.78	0.71	0.76	0.73

5. Conclusion and Future Work

This paper focused on a new approach for formulating question clusters by performing $(n\times (n-1)/2)$ pair-wise question vector comparisons. Similarity matrix computation was carried out using cosine similarity and jaccard similarity coefficient. Results obtained indicate that cosine similarity outperforms jaccard coefficient in formulating question clusters. The grouping strategy based Partition Algorithm has been found successful in reducing the time complexity O $(n\times (n-1)/2 \log n)$ used in hierarchical approaches to O$(n\times (n-1)/2$. The generated question clusters are very important in situations where novice instructors wish to formulate different sets of question papers for the same examination. The primary objective of this study was to identify the effectiveness of statistical measures in formulating question clusters. Our future work will focus on replacing the statistical approaches of similarity matrix generation by semantic approaches.

6. Reference

[1] M. Steinbach, G. Karypis, V. Kumar, A comparison of document clustering techniques, In KDD Workshop on Text Mining, 2000.
[2] K. Jain, M.N.Murty, P.J. Flynn, "Data clustering: A review", ACM Computing Surveys, 31(3):264–323, 1999.
[3] Robert A. Ekblaw, "Feasibility and effectiveness: Comparing document clustering algorithms from a user's perspective", Journal of Information Science, pp. 1-20, 2012.
[4] S. K. M. Wong and V.V. Raghavan, Vector space model of information retrieval: a reevaluation, In Proceedings of the 7th annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR '84, pp. 167-185, Swinton, UK, British Computer Society, 1984.
[5] Duc Thang Nguyen, Chee Keong Chan, "Clustering with Multiview point-Based Similarity Measure", IEEE transactions on knowledge and data engineering, vol. 24, no. 6, june 2012.
[6] Anna Huang, Similarity Measures for Text Document Clustering, In proceedings of New Zealand Computer Science Research Student Conference, NZCSRSC Christchurch, New Zealand, pp.49-56,April 2008.
[7] Yllias Chali, Soufiane Noureddine, "Document Clustering with Grouping and Chaining Algorithms", IJCNLP, Springer-Verlag Berlin Heidelberg, LNAI 3651, pp. 280–291, 2005.
[8] Slonim N., Tishby N., Document clustering using word clusters via the information bottleneck method, In Proceedings of the 23rd international conference on research and development in information retrieval, pp. 208–215, SIGIR 2000.
[9] Khairuddin N. N., Hashim, K., Application of Bloom's Taxonomy in Software Engineering Assessments, In Proceedings of the 8th WSEAS International Conference on Applied Computer Science, 2008.
[10] Krathwohl D. R., "A revision of Bloom's taxonomy: An overview, Theory Into Practice,Vol.41 (4), pp. 212-219,2002.
[11] Paice Chris D., An evaluation method for stemming algorithms, In proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval, pp.42-50, 1994,.
[12] Pascal Soucy, Beyond TFIDF weighting for text categorization in the vector space model, In Proceedings of the 19th International Joint Conference on Artificial Intelligence, 1130-1135, IJCAI 2005.
[13] Aizawa, A, "The feature quantity: an information-theoretic perspective of tf idf-like measures", In Proceedings of the 23rd ACM SIGIR conference on research and development in information retrieval, pp.104–111, 2000.