Clustering Examples in Web-based Tutoring Systems based on Relevance of Concepts

Ritu Chaturvedi
School of Computer Science
University of Guelph, Guelph, Canada
chaturvr@uoguelph.ca
C. I. Ezeife*
School of Computer Science
University of Windsor, Windsor, Canada
cezeife@uwindsor.ca

***Abstract*—Web-based online tutoring systems (WOTS) have become extremely important and relevant in today's world, especially with COVID-19 requiring schools, colleges and universities to offer alternate forms of delivery. Many studies have indicated that students find worked-out examples very useful, when they are performing a task or studying for final exams. WOTS certainly have the capability to host hundreds of such examples in their repositories, but presenting students with such repositories may cause cognitive overload on students and may force them to bear the responsibility of searching for the most relevant examples, when in need. This paper proposes an algorithm called CER (Clustering Examples based on Relevance) that organizes a collection of worked-out examples into coherent and relevant clusters - relevant to the learning concepts covered by them. When generating clusters, CER acknowledges not only the local relevance of a concept (using parameters such as mode) within a cluster but also its global relevance. The proposed algorithm CER is validated using Dunn's index as the internal validity index - a score of 0.81 was achieved for CER. The external validity of CER was measured by comparing its results to a benchmark dataset that had properties of data that were common to the domain of CER.**

I. Introduction

[1]Web-based online tutoring systems (WOTS) have become exptremely important and relevant in today's world, especially with COVID-19 requiring Universities and Colleges to offer alternate forms of delivery. One of the essential components of any WOTS is its domain model, that holds the learning outcomes of the course it teaches (also known as the domain), and more importantly, embeds the expertise in context with that domain - expertise in terms of lessons (in any format, be it text, audio or video), learning concepts covered by each lesson (LC), graded work assessments (such as tasks, assignments or projects), their solutions, and other helpful resources such as worked-out examples to help students learn and understand the domain LC. In traditional classroom teaching, such expertise comes from a combination of the teacher and the text book put together. The scope of this paper is limited to WOTS that teach "Introductory Programming in C".

This research was supported by the Natural Science and Engineering Research Council (NSERC) of Canada under an Operating grant (OGP-0194134) and a University of Windsor grant.

[1]978-1-7281-8084-7/20/$31.00 ©2020 IEEE

Clustering is an unsupervised mining method [Pang-Ning et al., 2005] that partitions a finite set of data points in multidimensional space into well-defined and separate clusters using distance measures such as Euclidean distance so that (1) data points belonging to the same cluster are similar (intra-cluster) and (2) data points belonging to different clusters are dissimilar (inter-cluster). For example, in an educational dataset, students can be grouped into two or more clusters according to their learning styles. A simple and effective clustering algorithm called k-means [Pang-Ning et al., 2005] (Algorithm 1) that takes as input an integer value k (where k = number of desired clusters) and n data points, where each data point is a vector with m m elements (e.g. each worked-out example in the domain of C programming is represented as a vector of m LCs). It then groups its points into k clusters (where each cluster consists of one or more data points) such that the inter-cluster similarity of the resulting clusters is low, whereas the intra-cluster similarity is high. Each group or cluster has a representative point known as its centroid or center. Intra-cluster similarity defines how close the points within a cluster are to each other, whereas inter-cluster similarity defines how well-separated the cluster centroids are from each other. A similarity or distance function (similarity is considered to be the inverse of distance) is used to find the closeness between a sample x and cluster centroid c or between 2 cluster centroids - functions such as Euclidean distance or Jaccard's Coefficient of similarity. The proposed algorithm CER (Clustering Examples based on Relevance) revises steps 2.1.1 and 2.2 of k-means (Algorithm 1) to accommodate for the relevance of concepts, both locally within a cluster and also in the entire global scope.

The rest of the paper is organized as follows. Section 2 outlines the proposed algorithm CER, with details on each step modified. Section 3 presents an experimental analysis and section 4 presents the conclusion, limitations of this work and future directions.

II. Knowledge Organization in WOTS using Clustering

This section presents the proposed algorithm CER, including the domain model, steps used for data preparation and the

Algorithm 1 K-means algorithm [Pang-Ning et al., 2005]

```
Input: dataset of size n X m (n samples, each with m attributes),
k (number of clusters),
maxIterations (threshold for maximum number of iterations)
Output: k clusters
Method
*** begin of k-means
1. Choose k samples as initial centroids
2. repeat until convergence (centroids not not change or maximum number of iterations has been reached)
    2.1. repeat steps until all n samples are exhausted
        2.1.1. assign sample x to its closest centroid using an appropriate distance function
    2.2. recompute the centroid of each cluster based on assignment in steps 2.1
*** end of k-means
```

final data representation used by CER to perform knowledge organization.

A. Domain Model

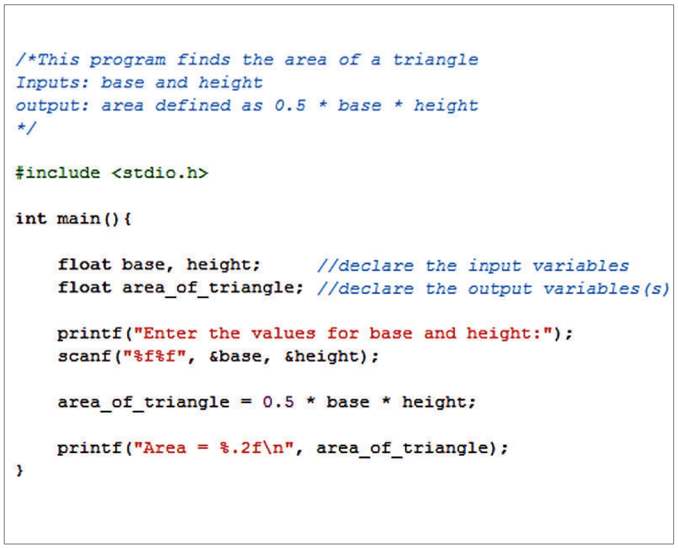
Domain model D_C for the proposed system consists of a repository of all worked-out examples designed by WOTS experts for a course that teaches “Introductory C Programming”. It also consists of all learning concepts (LC) in its domain (also referred to as a topics or concepts in this paper) that WOTS must teach in order to meet the learning outcomes of the course. For example, “scanf”, which is a command for entering values into variables from the keyboard, is a LC in the domain of C programming. Similarly, “fraction” is an LC in the domain of Math. The development of domain model is a very tedious job, and requires the time and effort of several domain experts [Chaturvedi et al., 2018].

A worked-out example (WE) in this paper is defined as a complete or partial worked-out solution for a question or instruction (similar to examples in textbooks). Figure 1 shows a sample worked-out example find_Area, which is essentially the solution to the following instruction: “Write a program that computes and prints the area of a triangle, given its base and height”.

A data point is a binary vector of size n, representing a worked-out example or a task solution in CER, where n is the total number of LCs in the domain of CER. For example, worked-out example find_area shown in figure 1 can be represented as a data point [1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], assuming m (total number of LCs in the domain) = 26.

B. Data Preparation and Representation

In an earlier research [Chaturvedi et al., 2018], [Chaturvedi, 2016], the authors used an extraction method that takes a WE and represents it as a vector of m binary values, where m is the total number of LCs in the domain. This paper chooses to use the same vector representation for each WE in its domain. The choice of a similarity function depends on the type of data attribute used - types such as continuous, categorical or binary. Continuous data attributes are those that can be measured (e.g. weight of a person). Categorical data attributes define different categories of data but cannot be measured (e.g. gender, that has two categories ’F’ and ’M’). Binary data attributes are a special case of categorical data, that can have only one of the two values 1 or 0. Binary data can be further categorized as symmetric and asymmetric data. A symmetric binary attribute is one in which the presence of a 1 is regarded as equally significant as its absence (0). An asymmetric binary attribute is one in which the presence of one of the values (e.g. 1) is regarded as more significant than the other. For example, in a vector of m binary values representing a WE, where a value of 1 indicates the presence of an LC and 0 its absence, then a 1-1 match of an LC in two different WEs is significant, whereas a 0-0 match has no significance (since 0 implies that the LU is not present) and must be ignored.



```
/*This program finds the area of a triangle
Inputs: base and height
output: area defined as 0.5 * base * height
*/

#include <stdio.h>

int main(){

    float base, height;     //declare the input variables
    float area_of_triangle; //declare the output variables(s)

    printf("Enter the values for base and height:");
    scanf("%f%f", &base, &height);

    area_of_triangle = 0.5 * base * height;

    printf("Area = %.2f\n", area_of_triangle);
}
```

Figure 1. Worked-out example find_Area.c as solution to the instruction ’Write a program that computes and prints the area of a triangle, given its base and height’

The most common similarity function used with binary asymmetric data is Jaccard’s coefficient (JC) [Jaccard, 1901] and therefore is is most applicable to the domain data used in this paper. Jaccard’s coefficient between two binary vectors x and y is measured as

$$JC(x,y) = \frac{f_{11}}{f_{11} + f_{01+}f_{01}} \tag{1}$$

where f_{11} is the frequency of occurrence of 1 and 1 in the corresponding bits of x and y, f_{01} is the frequency of occurrence of 0 and 1 in the corresponding bits of x and y and f_{10} is the frequency of occurrence of 1 and 0 in the corresponding bits of x and y. Here, f_{01} and f_{10} represent the non-matching attribute pairs. For example, if x = [1, 0, 0, 1] and y = [1, 0, 1, 0], then $JC(x, y) = 1/3$. , where m = number of elements in a data point.

C. Proposed Algorithm - Clustering Examples based on Relevance (CER)

CER focuses on organizing the list of worked-out examples in its domain based on the learning concepts (LC) they contain. The main motivation for proposing and implementing a learning concept based organization of worked-out examples is to enable WOTS to assist students to prepare for final examination, when they must have completed all other requirements in the course and must have learnt all the relevant learning concepts. The proposed CER algorithm will group all those worked-out examples that have related LCs into a single cluster (as opposed to traditional lesson-wise organization that we find in textbooks). For example, a cluster for "Introductory C Programming" that has examples on arrays in it, will also include examples with for-loops in it (CER experts assume that most C programs that use one-dimensional arrays tend to use simple for loops and those with two-dimensional arrays tend to use nested-for loops).

CER uses a modified k-means clustering algorithm to cluster its resources - the modifications are focused on including the relevance of LCs in any vector representation of a WE. It treats the problem of organizing worked-out examples into coherent groups or clusters as a problem of clustering binary datasets. There are existing algorithms that cluster binary symmetric data efficiently but do not consider the asymmetric nature of data. For example, Tao Li [Li, 2005] proposed a general framework for clustering binary data, in which they cluster both data and their features simultaneously using matrix approximation. They initialize k-means exactly as in standard k-means by randomly picking initial cluster centroids, but recompute the centroids using an alternating optimization method that minimizes the approximation error between original clusters and the recomputed ones. Their algorithm works well with symmetric binary data that is not too sparse, unlike the data used in this research. Similarly, Ordonez [Ordonez, 2003] proposed a modified k-means algorithm called Incremental k-means for symmetric sparse binary data. Incremental k-means uses Euclidean distance as the distance measure, which is meaningful only when data is measurable and its magnitude is significant (unlike the data used in this paper). The initialization step of Incremental k_means uses estimation maximization (EM) algorithm [Markov & Larose, 2007] to compute the initial set of cluster centroids but uses sum of euclidean distances to recompute centroids. The proposed CER algorithm uses not only the local information available to clusters (similar to k-means) but also global information on domain data as explained in the following section.

CER uses k-means as its core algorithm but with 2 important modifications - (1) step 1 of k-means is modified to choose a set of initial centroids using knowledge of the local neighborhood. To find each successive initial centroid, CER chooses the example that is farthest away from any of the already picked centroids. This guarantees that the initial set of centroids is well-separated. Section II-C1 presents this step as algorithm 3 (CER_{IC}). (2) step 2.2 of k-means is customized to recompute centroids for binary data. Conventional methods

Algorithm 2 CER (Clustering Examples based on Relevance)

```
Input: dataset $\partial$ of size m X n (m data points, each with n attributes), k (number of clusters
Output: k clusters
Method:
**** begin of CER
1. Choose k data points from $\partial$ as initial centroids using Algorithm 3 ($CER_{IC}$)
2. repeat until convergence
    2.1. repeat steps until all m data points are exhausted
        2.1.1. assign point x to its closest centroid using Jaccard's similarity (equation 1)
    2.2. recompute the centroid of each cluster using Algorithm 4 ($CER_{RC}$)
**** end of CER
```

such as computing the mean of all cluster members do not work well with binary data [Pang-Ning et al., 2005]. Mode, instead of mean is commonly used for categorical attributes including binary attributes but is not useful in the example dataset used in our research. Section II-C2 proposes a novel algorithm called CER_{RC}(algorithm 4) to recompute centroid bit for each LC to either 1 or 0, depending on the global relevance of the LC, in addition to mode, which is a local property of the cluster.

1) Neighborhood-sensitive choice of initial centroids (modified step 1 of k-means) : **Definition: Similarity vector :** Similarity vector sv between a data point e and all data points in a set S is defined to be a vector of size q, where q = number of data points in S . Each element of sv stores the Jaccard's similarity coefficient JC (equation 1) between e and each data point in S. Both e and data points in S must belong to the same domain and must have the same number of features.

For example, let q = 3, e = 1010, S = { s1: 1001, s2: 1101, s3: 0110}, sv =[0.5, 0.25, 0.33]

The choice of initial centroids impacts the number of iterations required to get the final clusters by k-means algorithms. Keeping in mind the primary goal of organizing worked-out examples into clusters based on the LCs they consist of, CER chooses as initial centroids those WE that consist of LCs that are very different conceptually, implying that they should be placed in different clusters. With this rationale, we choose a farthest-neighbor approach [Pang-Ning et al., 2005] to pick the initial set of centroids. CER chooses the first centroid randomly. Then, for each successive centroid, the worked-out example that is farthest away from any of the already chosen initial centroids is chosen using Jaccard's coefficient (equation 1). This guarantees that the initial set of centroid is well-separated. Algorithm 3 (CER_{IC}) that describes this method picks a worked-out example randomly as the first initial centroid (ic_1). Then it uses JC to generate the similarity vector sv (section II-C1) between ic_1 and all other worked-out examples in D_C. The worked-out example that has the least similarity value in sv (and therefore the farthest neighbor of ic_1) is chosen as the next centroid (ic_2). To find the third centroid (and the remaining ones), CER_{IC} finds the LCs that

Algorithm 3 CER_{IC} (modified step 1 of k-means)

```
Input : (1) binary matrix $\partial$ of m rows and n columns (each row of $\partial$ represents an example; each column of $\partial$ represents a LC in the example) ,
(2) desired number of clusters k
Output: matrix $ic$ of k rows and n columns, consisting of k initial centroids
Method:
***begin of $CER_{IC}$
1. add first row in $\partial$ to $ic[1]$—- each initial centroid is a vector of n 1s and 0s
2. compute similarity vector $sv$ between $ic[1]$ and $\partial$ using JC (equation 1).
3. find the example that has least similarity value in $sv$ and add that as the next centroid to $ic[2]$.
4. initialize z to 3.
5. repeat until desired number of clusters k is achieved (until z=k)
    5.1 $\partial$ = $\partial$ - {rows already stored in $ic$}
    5.2 compute $temp$ by assigning each bit $\mu_{ij}$ of temp to 1 or 0 using equation 2.
    5.3 compute similarity vector $sv$ between $temp$ and all rows in $\partial$ using JC (equation 1).
    5.4 find the example that has least similarity value in $sv$ and pick that as the initial centroid $ic[z]$; increment z
***end of $CER_{IC}$
```

are absent in the already chosen centroids (ic_1 and ic_2) and sets those bits to 1 in a temporary vector called $temp$ - we call each bit of $temp$ as μ_{ij} (jth bit of vector i) as defined in equation 2. Conceptually, $temp$ stores all LCs not already covered by the existing initial centroids. Thus, $temp$ is a good representative of the information on LCs covered by all the already chosen initial centroids.

$$\begin{aligned} \mu_{ij} &= 1, if\, \forall x \in \{ic\},\, l = \#\{ic\},\, \sum_{j=1}^{l} x_j < (l - \sum_{j=1}^{l} x_j) \\ &= 0,\, otherwise \end{aligned} \tag{2}$$

For example, if ic_1 = [0 1 0 0 0 1] and ic_2 = [1 0 0 0 0 1], then $temp$ vector = [0 0 1 1 1 0]. Next, it computes sv between $temp$ and the remaining examples in D_C to choose the example with the least similarity value as the next initial centroid ic. This process is repeated until the desired number of clusters is reached.

2) Recomputing centroids for binary data (modified step 2.1 of k-means): Step 2.2 of k-means recomputes the mean of all clusters to update the set of centroids, if necessary. For categorical attributes including binary ones, mode (instead of mean) of all clusters is commonly used to update centroids [Pang-Ning et al., 2005]. Mode of an attribute refers to the value that occurs most often. For example, let cluster c created from an iteration i of CER (Algorithm 2) has 7 data points in it, where each data point has 26 LCs as its features. If the first feature LC_1 for cluster c has values {1, 1, 0, 0, 1, 0, 1} for the 7 data points, then its mode is 1, since the number of 1s is more than the number of 0s. Therefore, the updated centroid's bit for LC_1 is assigned a value of 1. Conceptually, this implies that LC_1 is a good representative for this cluster. As this example illustrates, LCs that are contained in very few worked-out examples in the entire domain will suffer a bias, if mode is used to recompute centroid bits. For instance, in the above example, if feature LC_{26} for cluster c has values {0,0,0,1,1,0,1}, then using the mode method, centroid bit for LC_{26}= 0. But, if the total number of worked-out examples in D_C with learning unit LC_{26} is just 4, then cluster c has 3 out of the 4 worked-out examples that CER has with LC_{26}, and therefore, LC_{26} should be marked as a good representative of cluster c and its centroid bit should be assigned a value of 1 (instead of 0). We propose a novel algorithm CER_{RC} (algorithm 4) that uses a smart technique to assign a value of 1 to the new centroid bit in 2 situations: (1) when the mode is 1 (2) when the mode is 0 but the presence of that LC is more relevant globally. CER_{RC} recomputes centroids by first calculating the mode of feature LC_i, for all worked-out examples in its cluster. If the mode is 1 for an attribute LC_i, then it is considered a good representative for the cluster and its bit is set to 1. If mode is 0, then the frequency of occurrence of attribute LC_i, in the entire database (α_i) is determined and used to find the remaining number of worked examples with LC_i in them ($residual$). If the number of times LC_i appears in cluster c is at least a certain threshold number of the remaining examples ($residual$), then the new centroid's bit for LC_i is assigned a value of 1. After experimenting with different threshold values, we chose a value of 30%.

III. Experimental Analysis of CER

The proposed algorithm CER forms clusters of worked-out examples so that all examples belonging to a single cluster have related LCs. Clustering can be evaluated using both internal and external validation measures (also known as indices). Internal validity index (such as Dunn's index) is based on the information intrinsic to the data alone, whereas external validity index (such as f_score) is based on previous information about data. Internal validity index are preferred when the class of data used in clustering is not known in advance [Maulik & Bandyopadhyay, 2002]. Such indices are useful in validating the algorithms written for clustering, comparing them with others and in choosing the number of clusters. This paper chooses to evaluate CER using internal validity indices such as Dunn's index because (1) the objective of this experiment is to compare the performance of CER with standard k-means for binary data (2) CER does not have any pre-defined classes or clusters that can be used to compare the results of CER for external validation. Dunn's index [Dunn, 1974] for each cluster partition, given the distance $d(x, y)$ between two data points x and y is defined as

$$Dunn = min_{1 \le i \le k}\{min_{1 \le j \le k, i \ne j}\{\frac{\delta(D_i, D_j)}{max_{1 \le r \le k}\{\triangle(D_r)\}}\}\} \tag{3}$$

$$\delta(D_i, D_j) = min_{x \in D_i, y \in D_j}\{d(x, y)\}$$

$$\triangle(D_r) = max_{x,y \in D_r}\{d(x, y)\}$$

Algorithm 4 CER_{RC} (modified step 2.2 of k_means)

```
Input: (1) initial_centroids of k rows and n columns,
(2) example dataset of m rows and n columns E,
(3) k clusters,
(4) $threshold$
Output: k new_centroids
Method:
***begin of $CER_{RC}$
1. repeat for each cluster k
    1.1. repeat for each attribute i of k
        1.1.1. calculate its mode of all members of cluster k
        Let ones = number of 1s; zeros = number of 0s
            $mode = 1, if\ ones > zeros$
            $= -1, otherwise$
        1.1.2. if mode is equal to 1,
            1.1.2.1. assign new_centroid's bit i to 1
        1.1.3. else
            // $\alpha_i$ is the total number of examples in E that have i=1 in them
            1.1.3.1.$residual = (\alpha_i - ones)$,
            // cluster has at least threshold number of examples in E with $LC_i$
        1.1.4. if $ones >= residual * threshold$,
            1.1.4.1. assign new_centroid's bit i to 1
        1.1.5. else
            1.1.5.1. assign new_centroid's bit i to 0
***end of $CER_{RC}$
```

In equation 3, δ represents the inter-cluster distance between 2 clusters D_i and D_j. $\triangle D_r$ measures the intra-cluster distances in cluster D_r. Larger values of Dunn's index indicate good clusters, implying high intra-cluster similarity and low inter-cluster similarity. The value of k that maximizes Dunn is then chosen as the optimal number of clusters.

This section presents an experimental analysis of CER and compares its cluster formation with clusters formed by standard k-means algorithm (Algorithm 1) using different parameters such as Dunn's index, total number of iterations to converge and interpretability. Table I shows results of 4 different experiments done with CER dataset. This table indicates that Dunn's index is as high as 0.81 when CER is used for cluster formation as compared to standard k-means that gives a Dunn's index of 0.51. The optimal number of clusters formed by CER is found to be 6, when Dunn's index is the highest (Dunn's index = 0.81, when k = 6). We compute the Dunn's index index for all values of k from 1 to 15, and find that k = 6 has the highest value.

In order to further validate the proposed clustering algorithm CER, we implement it on a benchmark zoo dataset [Bache & Lichman, 2013] and compare its results with standard k-means. The reason for choosing this benchmark is the similarity in properties of data used in both zoo and CER. Zoo dataset [Bache & Lichman, 2013], available at the UCI Irvine Machine Learning Repository (uci.kdd) consists of 17 binary attributes and 1 categorical attribute. We transformed the non-binary attribute into 4 different binary attributes. Therefore, zoo dataset in our experiments consists of 101 instances (or records) of 21 binary attributes each (attributes such as hair, feathers, eggs and tail). There are 7 pre-defined classes of animals in the zoo dataset and the total number of animals in each class are 41, 20, 5, 13, 3, 8 and 10. For example, animals (frog, newt, toad) classify as class 5, whereas pitviper, seasnake, slowworm, tortoise, tuatara classify as class 3. CER, when applied to this zoo dataset, results in Dunn's index = 1.41 for 5 clusters (instead of 7). This can be attributed to the fact that classes 3 and 5 have very few instances in them (class 3 has just 5 animals in it and class 5 has 3 animals from a total of 101) and therefore CER is not able to identify them as separate clusters. Figure 2 shows the Dunn's index computed for all values of k from 1 to 15 for both datasets (CER and Zoo).

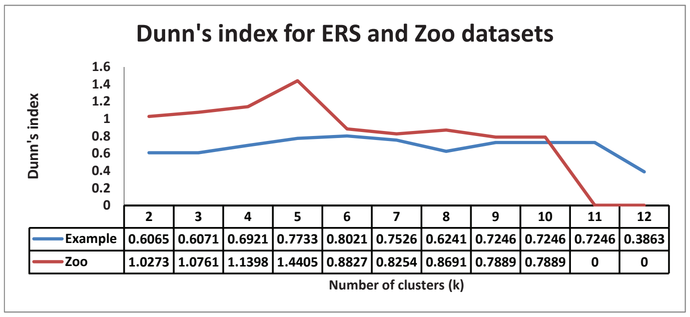
Case	Algorithm	Number of clusters	Number of iterations to converge	Dunn's Index
1	Modified Step1 Modified Step6 (Algorithm 2 - CER)	6	5	**0.802**
2	Modified Step1 Standard Step6	6	3	0.802
3	Standard Step1 Modified Step6	6	5	0.601
4	Standard Step1 Standard Step6	6	6	0.512

Table I
COMPARATIVE ANALYSIS OF PROPOSED MODIFIED K-MEANS WITH STANDARD K-MEANS - CER DATASET

We also validate the cluster formation in zoo dataset by CER against its actual pre-defined classes [Bache & Lichman, 2013] using external validation measures such as accuracy and f_score and use the results as a benchmark for CER (since both CER and zoo data are binary). A confusion matrix is generally created to compute such measures. A confusion matrix is a table that allows visualization of the performance of a classification algorithm. Each column of the matrix represents the instances in a predicted class, and each row represents the instances in an actual class. The confusion matrix in table II shows the total number of zoo classes predicted by CER that match the actual pre-defined ones. For example, row 1 in table II indicates that there are 36 (out of 41 animals in zoo dataset) that are predicted correctly as class 1, whereas there are 2 animals that are actually class 1 but are predicted incorrectly as class 2. Similarly, there are 3 animals that actually class 1 but are predicted incorrectly as class 5. Accuracy (given as total number of correct predictions / total number of predictions) for CER is found to be 77% and f_score is computed as 79%, whereas f_score using standard k-means is computed to be 74%. Details on computing accuracy and f-score are given in section. This comparison shows that the proposed algorithm CER is a viable algorithm to cluster data that is asymmetric and binary.

		P	R	E	D	I	C	T
	class	1	2	3	4	5	6	7
A	1	36	2	0	0	3	0	0
C	2	0	11	9	0	0	0	0
T	3	2	0	0	3	0	0	0
U	4	0	0	0	13	0	0	0
A	5	0	0	0	1	3	0	0
L	6	0	0	0	0	0	8	0
	7	0	0	0	1	0	2	7

Table II
CONFUSION MATRIX FOR THE BENCHMARK ZOO DATASET TO COMPARE THE ACTUAL ANIMAL CLASSES WITH THOSE PREDICTED BY CER



	2	3	4	5	6	7	8	9	10	11	12
Example	0.6065	0.6071	0.6921	0.7733	0.8021	0.7526	0.6241	0.7246	0.7246	0.7246	0.3863
Zoo	1.0273	1.0761	1.1398	1.4405	0.8827	0.8254	0.8691	0.7889	0.7889	0	0

Figure 2. Validity Indexes for k=1..12 for the benchmark Zoo and CER dataset

A. *Complexity of CER*

Complexity of CER is the same as that of standard k-means clustering. Complexity of CER to organize m number of worked-out examples with n number of LCs in each is $O(I * k * m * n)$ [Pang-Ning et al., 2005], where I is the number of iterations required to converge and k is the number of neighbors. Typically I is a small value and k is a value much smaller than m, therefore complexity of CER can be considered to be $O(m * n)$.

IV. CONCLUSION AND FUTURE WORKS

Web-based tutoring systems are seeing a major shift into completely online to mitigate issues of social distancing amidst the crisis of COVID-19. While a WOTS requires many different automated tasks to run successfully, this paper focuses on the task of helping students pick and study the most relevant worked-out examples from a collection that is organized based on the relevance of its domain's learning concepts. The proposed clustering algorithm CER is validated using Dunn's index as the internal validity index. A high value of Dunn's index indicates the formation of compact and well-separated clusters by CER. CER is also implemented on a benchmark binary dataset Zoo [Bache & Lichman, 2013] to evaluate if the number of clusters formed by CER are optimal. This is done by comparing these clusters formed by CER with predefined classes for Zoo as given in UCI Irvine Machine Learning Repository [Bache & Lichman, 2013] to compute an f_score of 79%.

Although the results of this study are promising, they still need to be validated with the student model component of any WOTS, so that usefulness of clusters generated by the proposed algorithm can be validated with real student interactions.

REFERENCES

[Bache & Lichman, 2013] Bache, K. & Lichman, M. (2013). Uci machine learning repository.

[Chaturvedi, 2016] Chaturvedi, R. (2016). Task-based example miner for intelligent tutoring systems.

[Chaturvedi et al., 2018] Chaturvedi, R., Brar, V., Geelal, J., & Kong, K. (2018). Concept extraction: A modular approach to extraction of source code concepts. In *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)* (pp. 1860–1866).: IEEE.

[Dunn, 1974] Dunn, J. C. (1974). Well-separated clusters and optimal fuzzy partitions. *Journal of cybernetics*, 4(1), 95–104.

[Jaccard, 1901] Jaccard, P. (1901). *Etude comparative de la distribution florale dans une portion des Alpes et du Jura*. Impr. Corbaz.

[Li, 2005] Li, T. (2005). A general model for clustering binary data. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining. ACM, 2005.* (pp. 188–197).

[Markov & Larose, 2007] Markov, Z. & Larose, D. (2007). *Data mining the web - Uncovering Patterns in Web Content, Structure and Usage*. John Wiley.

[Maulik & Bandyopadhyay, 2002] Maulik, U. & Bandyopadhyay, S. (2002). Performance evaluation of some clustering algorithms and validity indices. In *Pattern Analysis and Machine Intelligence, IEEE Transactions on ,*, volume 24 (pp. 1650–1654).

[Ordonez, 2003] Ordonez, C. (2003). Clustering binary data streams with k-means. In *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery. ACM, 2003.* (pp. 12–19).

[Pang-Ning et al., 2005] Pang-Ning, T., Steinbach, M., & Kumar, V. (2005). *Introduction to Data Mining*. Addison-Wesley.