A High-Performing Similarity Measure for Categorical Dataset with SF-Tree Clustering Algorithm

Mahmoud A. Mahdi
Faculty of Computers and Information
Zagazig University, Egypt

Samir E. Abdelrahman
Faculty of Computers and Information
Cairo University, Egypt
Department of Biomedical Informatics
University of Utah, USA

Reem Bahgat
Faculty of Computers and Information
Cairo University, Egypt

***Abstract*—Tasks such as clustering and classification assume the existence of a similarity measure to assess the similarity (or dissimilarity) of a pair of observations or clusters. The key difference between most clustering methods is in their similarity measures. This article proposes a new similarity measure function called PWO "Probability of the Weights between Overlapped items "which could be used in clustering categorical dataset; proves that PWO is a metric; presents a framework implementation to detect the best similarity value for different datasets; and improves the F-tree clustering algorithm with Semi-supervised method to refine the results. The experimental evaluation on real categorical datasets, such as "Mushrooms, KrVskp, Congressional Voting, Soybean-Large, Soybean-Small, Hepatitis, Zoo, Lenses, and Adult-Stretch" shows that PWO is more effective in measuring the similarity between categorical data than state-of-the-art algorithms; clustering based on PWO with pre-defined number of clusters results a good separation of classes with a high purity of average 80% coverage of real classes; and the overlap estimator perfectly estimates the value of the overlap threshold using a small sample of dataset of around 5% of data size.**

***Keywords*—Algorithm; clustering; similarity; measurement; categorical; F-Tree; SF-Tree**

I. Introduction

General data mining applications have two types of data, categorical and numerical. Most clustering algorithms focus on numerical data whose inherent geometric properties can be exploited naturally to define distance functions between data points [1]. Categorical data refers to the data describing objects, which have only categorical (non-numerical) attributes [2]. Such data is often related to transactions involving a finite set of elements, or items, in a common item universe [3]. Transactional data is a kind of categorical data in which records can have different sizes. It is generated by many applications such as e-commerce, healthcare, and CRM [4]. It plays an important role in many fields like market basket data, web usage data, customer profiles, patient symptoms' records, and image features. This paper focuses categorical and transactional data.

Clustering is a widely used technique in which data items are partitioned into groups (called clusters) based on their similarities or differences, such that data items in the same cluster are more similar among themselves than items in other clusters [5]. It is usually difficult to deal with categorical attributes; therefore, clustering of categorical attributes has not received as much attention as its numerical counterpart [6]. Categorical attributes have unique features from the definition in [2]; therefore, the traditional approach to convert categorical data into numerical values does not necessarily produce meaningful results specially in the case where categorical domains are not sorted [2], [7], [8]. For example, hierarchical clustering algorithms may be unstable when used to cluster categorical data because the distance between the centroid of clusters of categorical data is not a good estimator of the similarity between the data [9]. Partition clustering algorithms may also be unsuitable because the sets of items that define clusters may not have the same sizes since the cluster may contain a small subset of the possible number of items. Thus, it is possible that a pair of transactions in a cluster have few items in common [7]. Moreover, clustering categorical data involve complexity that is not encountered in numerical data. In addition, different clustering algorithms hardly generate the same clustering result for the same dataset. For these reasons, there is an unmet need for algorithms that tackle these limitations during clustering categorical data [6].

One of the most important aspects of data mining problem is how similarity measure is defined [10] and calculated [11], since the similarity measures have the effect of clustering and classifying information with respect to data types. Clustering techniques for categorical data are very different from those for numerical data in terms of the definition of similarity measure [12]. It is also rare to find the boundaries of the clusters and avoid overlapping between them, which adds an additional constraint to researchers when choosing the optimal similarity measure that could be applied to a wide range of data types. Most of the clustering algorithms have two phases: allocation and refinement phases. The refinement phase has two drawbacks: 1) its results depend on the results of the allocation phase; and 2) its run time complexity is relatively high. It is known that the size of transactional data is usually large, so there is a great demand for fast and high quality algorithms to cluster large-scale transactional datasets.

This article extends our prior study of measuring the similarity between clusters of categorical (or transactional) data in [13]. The list of this article contributions are presented in (Tables I and II) and are summarized as follows.

A criterion function is described in details for similarity measure called PWO (Probability of the Weights of Overlapped items) for categorical data to overcome the problem of overlapping between clusters. A new algorithm which depends on PWO is provided for clustering categorical datasets with possibly different dimensions. A new framework is proposed to estimate the best similarity threshold parameters for different datasets that could be used as the proposed clustering algorithm. The similarity measure is tested on real-world datasets obtained from the UCI Machine Learning Repository [14] and applied to find similar groups in models constructed from different datasets. Inferences from the similar groups found to be logically meaningful. Finally, the algorithm is also compared versus different state-of-the-art algorithms in terms of the purity, the number of clusters, and the performance.

To sum up, this article extends significantly the earlier work [13] in the aspects described in Tables (I, and II). Here, the PWO is discussed in details, propose the PWO as a standalone algorithm, improve F-Tree algorithm, and implement the overlap estimator algorithm. The algorithms are evaluated in details versus different algorithms using addition datasets.

The reset of this paper is organized as follows. Section 2 summarizes the general notation and definition. Section 3 discusses the related work to this paper. Section 4 discusses the PWO similarity measure that is used to calculate the similarity between clusters. Section 5 discusses the approach to cluster categorical data based on PWO similarity measure. Section 6 presents the overlap estimator framework to determine the overlap threshold. Section 7 describes F-Tree clustering algorithm. Sections 8 and 9 describe the data mentioned in this research followed by a comprehensive set of experiments and related discussions. Sections 10 and 11 present the limitations and conclusion of this study.

TABLE I. EXTENDED EFFORTS

	Prior paper [13]	This article contribution
PWO Measure	Summary	More focus and metric proof
PWO Algorithm		Novel
Overlap Estimator	Proposed idea	implementation
F-Tree Algorithm	More description	Summary, and add predefined number of clusters
SF-Tree Algorithm		Novel

TABLE II. EXTENDED EXPERIMENTS

	Prior paper [13]	This article contribution
PWO Measure		Evaluation of metric function
PWO Algorithm	Minimum support vs number of clusters	Evaluation and analysis clustering algorithm with(out) fixed number of clusters
Overlap Estimator		Analysis precision and scalability
F-Tree Algorithm	Compare with 4 algorithms	Compare with more than 10 other algorithms
SF-Tree Algorithm		Analysis with predefined number of clusters, without predefined number of clusters, and analysis minimum fit of training dataset
Dataset	Mushroom and Votes	Nine Datasets (Table IV)

II. NOTATION AND DEFINITION

In order to simplify the expressions throughout this paper, the following notations are used. Consider a categorical or transactional dataset D consisting of a set of transactions $\{t_1, t_2, ..., t_n\}$ of size N. where, each transaction T contains a set of items or attributes $I = \{i_1, i_2, ..., i_m\}$. Hence, Clustering $\{C_1, C_2, ..., C_k\}$ is a partition of transactions$\{t_1, t_2, ..., t_n\}$. Where, each C_k called a cluster and K is the total number of clusters. M_k, and N_k are used respectively to denote the number of distinct items, and the number of transactions in the cluster C_k. I_k represents the categorical items in a cluster C_k, where$I_k = \{i_{k1}, i_{k2}, ..., i_{kM}\}$. S_k is the sum of occurrences of all items in cluster C_k. Θ is the minimum support or the minimum number of item's occurrence that should be present in each cluster.

III. RELATED WORK

The recent categorical clustering techniques are reviewed in this section. Each algorithm follows one concept of the three main concepts. First, clustering algorithms based on a predefined knowledge of the number of clusters such as COOLCAT [15], LIMBO [16], Fast clustering [17], Ensemble [18], and Hybrid [19]. Second, clustering algorithms without any knowledge about the clusters such as LargeItem [20], SLR [21], SEED [22], CACTUS [23], CLOPE [24], CLICKS [25] and DELTA [26]. The last type includes clustering algorithms that depend on the number of clusters at further step in order to refine and improve the clustering or to have an ability to work in the first place, such as ROCK [7], WCD [4], Squeezer [1], and SCCADDS [27]. In addition, it have been found some authors presented many techniques to find the best number of clusters.

Most algorithms generate clusters in the allocation phase then try to refine them in the refinement phase depending on the similarity measure function. The numbers of refinement steps are then state, as they affect the algorithm's performance. Measure function is applied on either local clusters or global clusters or both. Approaches based on local function compute the evaluation function between items inside the same cluster; the result shows the degree of how items inside a cluster are related to each other. On the other hand, global approaches compute the evaluation function between clusters; the result shows the degree of how clusters are dissimilar and more distinct. Finally, the measurement parameters and their numbers are stated; increasing the number of parameters will increase the complexity of the algorithm and the difficulty of the user's experience.

The LargeItem [20] uses the concept of large items to divide the transactions into clusters. An item marked as large in a cluster of transactions if its occurrence rate is larger than a minimum support parameter that is specified by the user. The LargeItem approach scans each transaction and either allocates it to an existing cluster or assigns it to a new cluster based on a cost function. The process of choosing a cluster for each transaction is based on the global goodness of clustering. This goodness is measured by minimizing the total cost function. Therefore, the LargeItem algorithm needs to set two parameters the minimum support Θ and the large item factor or weight w. In addition, the LargeItem algorithm is exhaustive in the decision procedure of moving a transaction t to the best cluster. The data structure used to handle clusters

is complex, and the approach taken to update the criterion function is not efficient; although the implementation uses the B-Tree structure to increase the performance of updating, but it consumes a lot of memory in case of handling the large dimensions of a dataset with large number of attributes. Moreover, the procedure of scanning transactions one at a time in each refinement phase and writing it back to the file is very I/O consuming.

The ROCK [7] is based on the number of links between two records of data items, instead of the distances between them. The links capture the number of other records that the two are both sufficiently similar to it. ROCK heuristically optimizes a cluster quality function with respect to the number of links in an agglomerative hierarchical fashion. ROCK has proved to be quite effective in categorical data clustering, but it is naturally inefficient in processing large databases [24]. The base algorithm is cubic in the dataset size, which makes it unsuitable for large problems. Therefore, ROCK may be suitable for small datasets. The ROCK data's format assumes that the similarities between data items are given. Hence, ROCK uses the similarity measure between two transactions as a number of common neighbors, but the computational cost is heavy and sampling has to be used when clustering large dataset [7]. The choice of $f(T)$ is critical in defining the fitness function, and the authors point out that the function depends on the dataset as well as on the kind of clusters that the user is interested in. Thus, the choice of the function is a weak and difficult task [15]. Besides, ROCK is difficult to fine-tune to find the right parameter T.

The COOLCAT [15] algorithm is based on the idea of entropy reduction within the generated clusters. Therefore, it does not rely on distance on arbitrary metrics. The algorithm groups points in the dataset trying to minimize the expected entropy of the clusters. This approach requires only parameter, which makes it stable and useful for larger datasets. However, the problem appears in the order in which the points are processed or grouped because of the point that appears to be a good fit for a cluster using a particular order of process may become a poor fit as more points are clustered using another order of process. To reduce this problem, the author added a re-processing step of a fraction of the points in the batch, so points are clustered in each batch and the worst fit points are re-clustered, while the number of occurrences for each of the attributes' values in a particular cluster is used to determine the goodness of the fit. However, this step increases the complexity of the algorithm specially in determining the number of fractions and worst fit points.

The CACTUS [23] is based on the concept of the common occurrences for the categories of different variables. The categories are considered strongly connected if the difference in the number of occurrences is greater than a user-defined threshold. The algorithm includes three phases: summarization, clustering and verification. In the summarization phase, the summary information is computed from the dataset. In the clustering phase, the summary information is used in discovering a set of candidate clusters. In the validation phase, the actual set of clusters is determined from the set of candidate clusters [6]. CACTUS could perform better, if the inter-attribute and intra-attribute summaries fit in the main memory. Like ROCK, this algorithm may be more suitable for small datasets. There are main problems with CACTUS; first, it does not scale since it requires the calculation and storage of potentially large similarity matrices; second, it lacks stability when the data is re-shuffled in the similarity matrices, it includes an unnatural distinguishing set assumption; and third there is no extension step after the cluster projections is found.

Small-Large Ratio or SLR [21] uses the measure of the ratio between small to large items; the item is marked as large or small depending on the number of its occurrences in a cluster. The algorithm tries to minimize the ratio of the number of small items to that of large items in each cluster. The goal of this method focuses on designing an efficient algorithm for the refinement phase of the LargeItem algorithm [20]. The SLR algorithm compares the small to large items' ratios with the pre-specified SLR threshold α to decide the best cluster for each transaction. SLR needs to set the support Θ, the weight w, the maximal ceiling E, and the SLR threshold α. In general, the SLR algorithm must compute all the costs of clustering when transaction *t* is put into another cluster to use the small-large ratios, which adds additional computational steps, and the large number of parameters makes it difficult to adapt the algorithms. Thus, the algorithm did not reduce the memory and I/O consumption of the LargeItem method. The authors in [22] concluded that both the LargeItem and SLR method suffer a common drawback; that they may fail to give a good representation of the clusters.

The CLOPE [24] approach depends on the ratio between the height and the and width. The height represents the number of transaction's item occurrence, while the width represents the number of clusters. The CLOPE tries to increase the height-to-width ratio of the cluster histogram. The larger height-to-width ratio of histogram the better intra-cluster similarity. CLOPE needs to set the repulsion r. There are two disadvantages of the r parameter. First, if there is no knowledge about the behavior of the dataset, it is difficult for users to expect the best value of repulsion r [4], so they must run the clustering phases more than once to get feedback on the best values for the clusters' numbers. Second, the CLOPE algorithm runs slower for non-integer repulsion r-values because of the computational overhead that comes with the floating point. The algorithm also requires two additional steps to handle adding and removing a transaction into and from a cluster in case of any refinement step.

The SEED [22] approach generates an initial seed of cluster centroid. The algorithm starts by finding the optimal number of clusters. SEED tries to maximize the fitness function value. This fitness measure calculates the average similarity between every transaction in a cluster to its centroid. The update of centroids will result in the need for clusters' re-organization. The process of centroid update and clusters' re-organization will be repeated until a suitable point of stability of the fitness function is reached.

The WCD [4] algorithm tries to preserve as many frequent items as possible within clusters and controls items' overlap between clusters. The WCD uses a partition-based clustering approach and tries to maximize the criterion function EWCD, "Expected Weighted Coverage Density". However, by default, when all transactions are considered in a single cluster, it will get the maximum EWCD, since this function cannot determine when the algorithm has to stop because merging

clusters maximizes the EWCD. Therefore, an additional phase prior to the clustering phases is required to determine the best number of clusters by taking a sample of data and running it on different values of K. This makes the algorithm's performance poor in a dynamic environment since the number of clusters can suddenly change.

The CLUC [28] algorithm depends on a similarity measure called cohesion that determines the degree with which items belong to clusters. The CLUC clusters data in two phases, initialization and refinement as most of the algorithms. But, this approach is most similar to LargeItem [20] except in the way of assigning the items in each cluster. The main drawback is found in multiple scanning of data to complete the clustering process.

The LIMBO [16] algorithm is based on the Information Bottleneck (IB) method, which uses the mutual information metric to define the measure for categorical clustering. Therefore, the algorithm works to minimize the information loss when grouping the items into clusters. The clustering approach works with three phases, Building Tree, Clustering, and associating tuples with clusters. The benefit of this approach is that it uses to cluster both tuples and attribute values; therefore, it can be classified as a hybrid algorithm.

The CBDT [29] approach is based on the distance between transactions. In this approach, the similarity between clusters is processed in three stages of calculating the distances: 1) between individual items; 2) between corresponding cells of different transactions; and 3) between transactions themselves. Therefore, the similarity measurements is considered time consuming. This is in addition to the large memory needed to store the pairwise results, which makes the algorithm's performance poor in large-scale data.

The Squeezer [1] approach works with one phase and is most similar to the allocation phase of the LargeItem algorithm with a similarity measure based on statistics. The algorithm reads transactions data in sequence and assigns it to the first maximum similar cluster or assigns it to a new cluster based on a minimum similarity. The output from this approach could change in case of change in the sequence of data input, as there is no refinement step. In addition, the similarity measure result will depend on the first clusters generation. The performance of this algorithm is better for uses with large-scale data; however, the approach use only local similarity computations to determine the maximum similar cluster.

Table III presents a comparison between the studied algorithms in the above. The clustering approach of each technique and the clustering phases are stated, as they affect the purity as well as the number of clusters.

IV. Proposed PWO Similarity Measure

The goal of any clustering algorithm is to reach the final pure state of clusters, so an estimation function must be adapted to measure how many object in one cluster are different with each objects in other clusters and at the same time, the objects within a cluster are similar. It is noted that the key difference between most of the methods is in defining the criterion function for measuring similarity. However, the difficulty lies in proposing a good scenario to solve the overlapping between clusters that depends on the underlying dataset. Therefore, to solve the overlapping problem, there are two requirements need to be considered in evaluation: the maximization of the frequent items within clusters and the minimization of the items that are overlapping between clusters.

A. The Overlapping Weight

It refers to the number of occurrences of an item in a cluster C as the item weight in C. For the purpose of the comparisons with the proposed measure function, the modified Jaccard theory [30] of similarity between clusters is described as in (1).

$$J_c(C_1, C_2) = \frac{|C_1 \bigcap C_2|}{|C_1 \bigcup C_2| - |C_1 \bigcap C_2| + 1} \tag{1}$$

In that sense, the similarity between two clusters increases along with the increase of the total intersection between them comparing with the total difference of the intra-join. It is noticed that Jaccard similarity neglects the weight (or support) of items in the clusters, which is significant in case of cluster's categorical dataset. Therefore, another measure that takes the weight of items is needed to take in consideration when measuring similarity, not only the number of items in the intersection.

B. The Probability of the Weights of Overlapped Items

The Probability of the Weights of Overlapped items (PWO) is introduced as a new measure function that estimates the goodness of clusters. Given a cluster C_k, suppose the number of distinct items is M_k, the items set of C_k is $I_K = I_{k1}, I_{k2}, \ldots, I_{kM}$, and the sum of occurrences of all items in cluster C_k is S_k, as calculated by (2).

$$S_k = \sum_{j=1}^{M_k} |I_{kj}| \tag{2}$$

Now, the weight of an item,W_{Ij}, inside a cluster C_k is defined as the ratio of occurrences of an item I_j to the sum of occurrences of all items inside the cluster; in other words the probability of an item inside the cluster C_k, is as shown by (3).

$$W_{Ij} = P(I_j) = \frac{|I_j|}{S_k} = \frac{|I_j|}{\sum_{j=1}^{M_k} |I_{kj}|} \tag{3}$$

In this sense, the total probability of all items within a cluster is equal to one, (4).

$$\sum_{j=1}^{M_k} W_{Ij} = \sum_{j=1}^{M_k} P(I_j) = 1 \tag{4}$$

Let overlap O_{ij} be the list of mutual items between cluster C_i and cluster C_j, where $O_{ij} = C_i \bigcap C_j$, and $|O_{ij}|$ represents the number of mutual items. All items belonging to O_{ij} have two possible weights: its weight for each cluster separately, and, its weight depending on the group of transactions.

TABLE III. SUMMARY OF CATEGORICAL CLUSTERING ALGORITHMS

Algorithm	Clustering Approach	No. of Phases	Measurement Approach	Metric Parameters	No. of Parameters	No. of Classes
Squeezer	Assign data to the first max similar cluster.	1	Local Similarity	Minimum similarity s	1	N/Y
SLR	Minimize the small large Ratio between clusters	2	Local and Global Similarity	Minimum support Θ, weight w, maximal ceiling E, and the SLR threshold α	4	N
LargeItem	Increase items' frequency inside clusters			Minimum support Θ, and the weight w	2	N
CACTUS	Depend on shared items between clusters			Distinguishing number K, passes D		N
COOLCAT	Minimize the expected entropy for each clusters			Minimum Entropy	1	Y
CLOPE	Increasing the high-to-width ratio of the cluster histogram			Repulsion r		N
ROCK	Increase the number of links between items inside the cluster			Fitness function $f(T)$		N/Y
CLUC	Depend on the cohesion measuring similarity to assigned items to clusters			User-defined threshold α		N
CBDT	Pairwise less distance between transactions	3		Number of classes r		Y
WCD	Increase the coverage of large items inside the cluster			Number of classes K		Y
LIMBO	Minimize the information loss.			Information loss threshold α		Y
SEED	Generated of an initial seeding of cluster centroids		Global Similarity	Minimum support Θ		N

Now, let $WO(C_i \mid C_j)$ represents the sum of weights of all items of a cluster C_i that overlap or intersect with cluster C_j, and expressed in (5).

$$WO(C_i \mid C_j) = \sum_{k=1}^{|O_{ij}|} W_{I_k \in (C_i \bigcap C_j)} \tag{5}$$

Similarly, $WO(C_i \mid C_j)$ is the sum of weights of all items of cluster C_j that overlap with cluster C_i. Now the definition of the probability of the weights of overlapped items between clusters C_i and C_j, $PWO(C_i, C_j)$, is presented as (6).

$$PWO(C_i, C_j) = WO(C_i \mid C_j) \textbf{.} WO(C_j \mid C_i) \tag{6}$$

Hence, the similarity measure is defined by (7).

$$sim(C_i, C_j) = PWO(C_i, C_j) \tag{7}$$

Now, the similarity function $sim(C_i, C_j)$ captures the closeness between the pair of clusters C_i and C_j. Actually, the sim values range between zero and one, with larger values indicating that the clusters are more similar. The sim value is one for identically matching clusters and zero for very dissimilar clusters.

C. Proof of PWO Similarity Metric

In [31] defines the similarity measure S as a function with a non-negative real value that satisfies three properties (1-3). Moreover, if S satisfies properties (4) and (5) then S is called a metric similarity measure.

1) $\exists s_0 \in R : -\infty < S(x, y) \leq s_0 < +\infty, \forall x, y \in X.$
2) $s(x, x) = s_0 \forall x \in X.$
3) $s(x, y) = s(y, x) \forall x, y \in X.$
4) $s(x, y) = s_0 \leftrightarrow x = y s(x, y).$
5) $s(y, z) \leq [s(x, y) + s(y, z)] s(x, z) \forall x, y, z \in X.$

The following is a proof of PWO being a metric similarity measure. Proof. $S = S_{PWO}$:

1) Property 1 is satisfied by the properties of probabilities $0 \leq S_{PWO} \leq 1$, thus $s_0 = 1$.
2) Property 2 is trivially satisfied by the fact that S_{PWO} is a similarity measure, thus $S_{PWO}(x, x) = 1$.
3) Property 3 is also satisfied by the fact that S_{PWO} is a similarity measure, so $S_{PWO}(x, y) = S_{PWO}(y, x)$.
4) If $S_{PWO}(x, y) = 1$ then x and y are identical, which together with (2) satisfies Property 4.
5) Property 5 is also satisfied by the properties of probabilities.

V. PWO CLUSTERING

A. Clustering using PWO

In order to study the ability of PWO as a similarity measure in clustering, the cost function in the LargeItem algorithm [20] and the PWO similarity metric are compared. While the goal of LargeItem is to minimize the total cost of each cluster, the goal is to maximize the similarity between transactions in the same cluster. Therefore, the LargeItem algorithm it modified to adapt this goal. Another important point is that the cost function of LargeItem is relative to the $MinimumSupport(\theta)$ that is given by the user. Therefore, the similarity between a pair of clusters is only accepted if it is larger than or equal to the minimum support, as shown in (8). Thus, higher values of θ correspond to higher thresholds for the similarity between a pair of clusters before they are considered similar. Algorithm.1 illustrates an overview of the clustering algorithm.

$$sim(C_i, C_j) \geq Minimum - support(\theta) \tag{8}$$

B. Merging Clusters by Groups

To speed up the clustering process, a new strategy of merging clusters is applied. Instead of merging the most

Algorithm 1 PWO Clustering Algorithm

```
while not EndOfFile do
  Read the next transaction < t, − >
  Allocate t to an existing C_i with MAX similarity larger
  than MIN support or in a new cluster
  Write < t, C_i >
end while
{Refinement phase}
not_moved ← true {not_moved true if no transaction t is
moved between clusters.}
repeat
  while not EndOfFile do
    Read the next transaction < t, C_i >
    Move t to an existing non-singleton cluster C_j that has
    a MAX similarity with it
    if C_i ≠ C_j then
      Write < t, C_j >
      not_moved ← false
      eliminate any empty cluster
    end if
  end while
until not_moved
```

Algorithm 2 PWO-M Clustering Algorithm

```
while not EndOfFile do
  Read the next transaction < t, − >
  Allocate t to an existing C_i with MAX similarity larger
  than MIN support or in a new cluster
end while
{Refinement phase}
no_merge ← false
{not_merge: false if no similar cluster found to merge.}
repeat
  num_cluster ← i;
  group clusters with similarity ≥α;
  if num_cluster == i then
    no_merge ← true
  end if
until not_merge == true
```

similar pairs of clusters, the most similar groups of clusters are merged. First, each transaction is allocated in a single cluster and compute the similarity between clusters, and then each cluster is assigned to a group if it is similar to any cluster in the group. The similarity's rule is applied using (8). The process of merging group of clusters at once instead of merging pairs of clusters minimizes the number of merging steps without affecting the purity.

Algorithm 2 presents a fast way of clustering transactions with the use of PWO as the similarity measure as well as in group's merging. To differentiate it from the previous algorithm explained in Algorithm 1, this algorithm is called PWO-M, in which groups of similar clusters continue to be merged until no more merge is possible. This approach speeds up the clustering process while maintaining the same degree of purity and number of clusters.

Algorithm 3 PWO Clustering Algorithm with Predefined Number of clustering

```
while not EndOfFile do
  Read the next transaction < t, − >
  Allocate t to an existing C_i with MAX similarity larger
  than MIN support or in a new cluster
end while
{Refinement phase}
repeat
  group pair clusters with MAX sim(Ci, Cj)
until num_cluster == predefined
```

C. Clustering using a pre-defined number of clusters

In case there is a pre-defined number of clusters before clustering, it can modify the algorithm presented in Algorithm 2 can modified to be as the one presented in Algorithm 3 by adding one additional step in the refinement phase. Merging pairs is continued with maximum similarity until the number of clusters reaches the required number. Results of this modification presented in the Experiments.

VI. Overlap Estimator

The similarity of clusters changes according to the dataset, so the overlapping between clusters varies depending on the behavior of the dataset. An automated framework is proposed to specify the best threshold value for determining the cluster's neighbor, i.e. similar clusters, according to the training dataset.

A. Cluster's Neighbours

Initially, A cluster's neighbors are those clusters that are considerably similar to it, and therefore they can be merged with it forming a large cluster.

Now, Let $sim(C_i, C_j)$ be a similarity function that normalizes and captures the degree of similarity between the pair of clusters C_i and C_j. The sim values are between zero and one, with larger values indicating that the clusters are more similar.

Given a threshold α between 0 and 1, a pair of clusters C_i, C_j are defined to be neighbors if (9) holds

$$sim(C_i, C_j) \geq \alpha \tag{9}$$

In (9), α is a parameter that can be used to control how close a pair of clusters must be in order to be considered neighbors; it is called the "neighborhood threshold".

Accordingly, higher values of α correspond to higher thresholds for the similarity between the pair of clusters before they are considered neighbors. Assuming that sim is one for matching clusters and zero for very dissimilar clusters, a value of one for α constrains clusters to be neighbors to only identical clusters. On the other hand, a value of zero for α permits any arbitrary pair of clusters to be neighbors. Equation (8) is the same as (9) if $\alpha = \theta$; θ is a user parameter while α is the clusters' neighbor threshold.

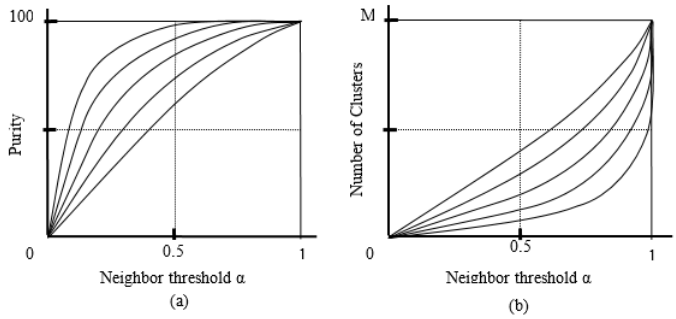

Fig. 1. The Neighbour's threshold relationship with purity and number of clusters.

B. Characteristics of the Neighbor's Threshold

The neighbor's threshold α has the following characteristics:

- Increasing (decreasing) the neighbour's threshold α increases (decreases) purity and increases (decreases) the number of clusters.
- Its result depends on the data type. A value of α that produces a purity of 100% on data D_1 does not necessarily produce a purity of 100% on data D_2.

Fig. 1 summarizes the relationship between the neighbor's threshold from one side, and purity and number of clusters from another side. In addition, it represents different dataset type curves. The growth rate with purity is appearing logarithmically in Fig. 1(a), while it is exponential with the number of clusters as shown in Fig. 1(b).

C. Estimating the overlap parameter

The best value of the neighbor's threshold α is the value that would get high purity with minimum number of clusters. The best value is called the overlap threshold α. The overlap estimator is used to estimate the best value of the neighbor's threshold α. The framework tries to find the minimum value of closeness of cluster's neighbor that will produce 100% purity of clusters, and uses this value in clustering the transactions of dataset.

To estimate the best overlap threshold α value, the following approach is applied:

1) Starting by clustering the training dataset at a minimum support of 100%.
2) Merging the clusters using different values of the neighbour's threshold α.
3) Test the purity of the output clusters based on the training set classes values.
4) Repeating steps (2-3) until reaching the minimum value of the neighbour threshold α getting 100% purity value.

In addition, it is noticed that starting from $\alpha = 0$ ascending to one is faster than starting from $\alpha = 1$ descending to zero in computing the similarities. Algorithm 4 shows the overlap estimator algorithm.

Algorithm 4 The Overlap Estimator Algorithm

Ensure: Input: The training dataset D_n, Classes C_n
  $\alpha \leftarrow 0$ {α is the threshold paramter}
  while Purity!=1 **do**
    $\alpha \leftarrow \alpha + 0.1$
    PWO-M (D_n, α, Output)
    Check(C_n, Output, Purity)
  end while
  return $\alpha - 0.1$

VII. F-Tree Clustering

Unlike traditional data, categorical clustering requires transactions to be partitioned across clusters in such a manner that instances within a cluster share a common set of large items, where the concept of the large follows the same meaning attributed to frequent items in association rule mining [32]. Thus, it is clear that categorical clustering requires a fundamentally different approach from the traditional clustering technique. F-Tree [13] is a summarization clustering algorithm that clusters categorical data based on a new tree structure.

A. F-Tree Clustering Algorithm

The basic F-Tree approach consists of the four main steps as follows:

1) Calculating items' frequencies: it scans the input dataset to rank all items.
2) Building a F-Tree: it inserts all transactional items of the dataset into F-Tree structure; it uses the frequencies of items to reorder the transactional items before inserting it into the F-Tree as discussed in the previous example.
3) Extracting initial clusters: initial clusters are generated using F-Tree by pruning the F-Tree at some level based on the minimum support.
4) Refining clusters: it applies the merging algorithm operated with PWO measurement to merge similar clusters that are extracted from the previous step.

All the above steps are divided in two phases. The allocation phase is concerned with the first three steps; while the refinement phase is concerned with only the last step.

1) F-Tree Data Structure: The F-Tree data structure is designed to compress the categorical dataset. As the categorical dataset contains a set of records, the F-Tree groups the records using their shared items or attributes in the tree. In the beginning, the global item frequencies are computed. Then all transactions' items are inserted in the F-tree using the item as the node key. The insertion of item is based on the global frequency order. As a result, the groups of items that get in the path starting from the tree root to any leaf node composes a single record, and so all paths from root to leaves nodes compose all transactions in the dataset. Thus, each sub-path can represent a set of transactions, which share the same path prefix. The following example illustrates the F-Tree process.

Suppose the following is the transaction records {ACF, ACH, BDE, AE, BF}. Computing the global items frequency gives {A(3), C(2), B(2), F(2), E(2), D(1), H(1)}. Now sorting the records' items based on global frequency gives {ACF,

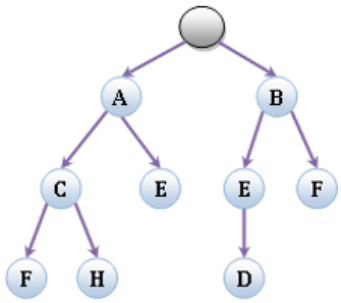

Fig. 2. An example diagram of an F-Tree structure.

ACH, BED, AE, BF}. Fig. 2 shows the F-Tree structure of these records; then, it can absolutely be seen that node A is a shared item between three records {ACF, ACH, AE}. In addition, Node B is a shared item between two records {BED, BF}.

2) Generate Clusters from F-Tree: Extract the clusters from the F-Tree depending on the F-Tree levels since each inner node shares all items in the upper level. Hence, the minimum support parameter is replaced by F-Tree depth, and for generalization, (10) is used.

$$ClusterLevel = MinimumSupport * tree_{depth} \quad (10)$$

To illustrate how the clusters are extracted from the previous graph in Fig. 2, for instance if the minimum_support$\theta = 75\%$, and $tree\_depth = 3$ then the cluster level = 2. At the second level (depth) there are four nodes. Each of those nodes contains one or more transaction in it or in its children. Therefore, there are four clusters as {{ACF, ACH}, {AE}, {BED}, {BF}}.

While if extracting clusters at the first level (depth), there are only two clusters as well as there are two nodes. These clusters are {{ACF, ACH, AE} , {BED, BF}}. At this point, it easy to notice that the number of clusters decreases as the level goes up to the root of the tree.

3) Merging Clustering Algorithm: The major steps of the merging algorithm are defined as the following:

- Computing the similarity list between clusters.
- Creating the group of neighbour clusters.
- Merging all clusters in the same group.
- Repeating these steps until there is no further merging.

The similarity list between clusters are computed using the merging overlap threshold α. Then, any similar cluster will belong to the same group in which the group of clusters contains only neighbour pairs of clusters. Lastly, it merges all clusters' neighbours inside the same group. A new generation of clusters could be more likely similar or dissimilar based on the result of merging. Therefore, the refinement procedure will repeat the merging algorithm until there is no more merge done or no new cluster's neighbour found.

Algorithm 5 The SP-TREE Algorithm

Ensure: The dataset D_n, Training set S_m
F-Tree ($D_n + S_m$, α, C_i)
for all s Cluster in Seed S_m **do**
Merge all clusters c contains any items $\in s$
end for
return Clusters C_k

B. Semi-Supervisor F-Tree Clustering

The F-Tree generates a large set of pure clusters, and although the refinement step breaks down the number of generated clusters, the refinement step needs a learning process to minimize the number of pure clusters without losing its precision specially when there is a predefined number of clusters. So, a training data base could be used as a seed in the merging phase. These seeds will be used to guide the merge algorithm with correct similar sets to speed up and correct the fitting of the merging algorithm. Algorithm 5 shows the ST-Tree algorithm.

VIII. Experiments

In this section, the accuracy is analyzed, precision and execution time of the proposed measure metric and clustering algorithms with real-life datasets. Several experiments are conducted for clustering to evaluate the general purity of the clustering algorithm using PWO. The version of LargeItem [20] algorithm is implemented for comparing performance and precision of clustering purpose, as well as LargeItem algorithm is a tree based techniques. Others algorithms results are collected from authors references.

A. Empirical Datasets

labeled datasets in Table (IV) are obtained from the UCI Machine Learning Repository [14] are used. The number of clustering presented in this table are used in clustering evaluation.

B. Pre-processing The Dataset

The input dataset is converted into a format that can be processed by algorithms handling transactional datasets. First, the class label is removed from all datasets. Second, each record of dataset is converted to a list of distinct items by mapping each property character for any attribute to a distinct numerical number for all, since each record of transactional data always contains a list of distinct items, and the record of a dataset has a list of properties' characters that may be duplicate throughout different attributes.

For instance, if there are the following dataset record {{F, G, F, F, F}, {F, G, T, N, F}, {Y, N, T, N, F}} with five attributes, then after mapping the transaction record would be {{1, 3, 5, 7, 9}, {1, 3, 6, 8, 9}, {2, 4, 6, 8, 9}. The same mapping process is applied if the dataset contained Boolean values. For instance, if the dataset record is {{1, 0, 0, 1, 0}, {1, 0, 0, 1, 1}, {0, 1, 1, 0, 0}}, then the equivalent transaction record would be {{1, 3, 5, 7, 9}, {1, 3, 5, 7, 10}, {2, 4, 6, 8, 9}}.

TABLE IV. DATASET PROPERTIES

Dataset	Size	No. of classes		No. of features	Total Attribute	Missing Values
		Num.	Cat.			
Mushroom	8124	2	1	22	23	2480
Chess	3196	2	0	36	36	0
Car	1728	4	0	6	6	0
Pima diabetes	768	3	8	0	8	0
Breast cancer	699	2	0	9	9	0
Vote	435	2	0	16	16	288
Wine	178	3	13	0	13	0
Iris	150	3	4	0	4	0
Zoo	101	7	1	15	16	0

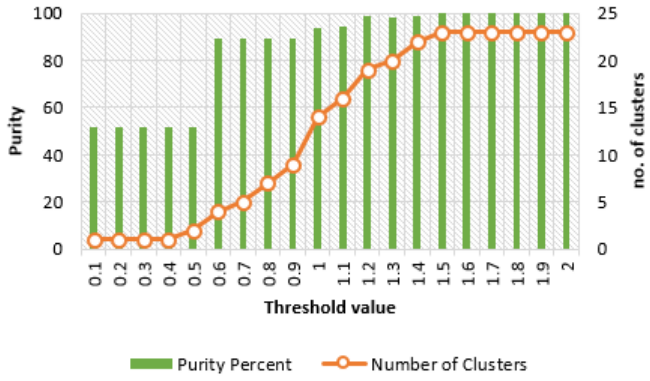

Fig. 3. The effect of threshold value on the data purity after merging clusters with Jaccard metric on the mushroom dataset.

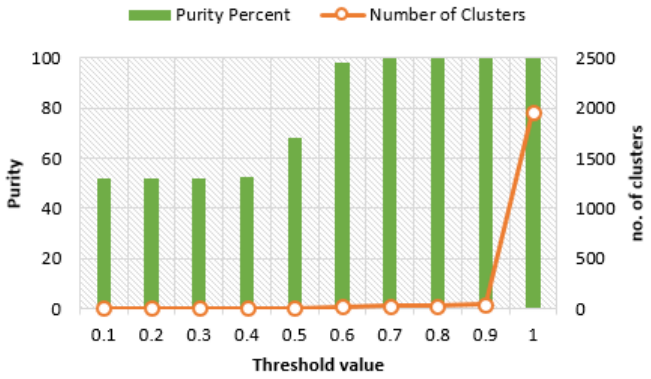

Fig. 4. The effect of threshold value on the data purity after merging clusters with PWO metric on the mushroom dataset.

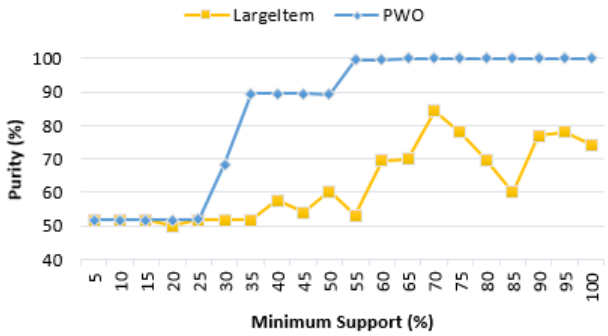

Fig. 5. Clusters Purity Results LargeItem vs. PWO on the mushroom dataset.

IX. EVALUATION STUDY

A. PWO Evaluations

1) Experiment 1: Evaluation of PWO Metric Function: To illustrate the advantage of PWO metric function, an analytical test is performed, in which the PWO and Jaccard capability are compared in grouping similar clusters. First, the base clusters are generated, then the similar clusters are computed using the Jaccard coefficient and then the PWO metric to merge the similar clusters in groups. Similar clusters are determined if the metric result is above the threshold value. Fig. 3 and 4 shows the result of purity with number of clusters using the Jaccard coefficient and PWO metric respectively applied on the mushroom dataset. In Fig. 4, the "100%" purity is reached when threshold value of Jaccard is equal to 1.5 and the number of clusters is "23". While in Fig. 4, the "100%" purity is reached at threshold value of PWO equal to 0.7 and the number of clusters is 23. Comparing between the two figures indicate the normalization strength of PWO to measure the similarity.

2) Experiment 2: Evaluation of PWO Based Clustering Algorithms: In this experiment, the LargeItem algorithm is modified by changing its cost function with PWO.

First, the purity of resulting clusters are compared from both algorithms on the mushroom dataset. Fig. 5 shows the result of this test, and it is observed that PWO reaches purity of 90% with minimum support of 35%. As a conclusion it is found that PWO is a powerful clustering similarity measure.

Second, the resulting number of clusters is compared. It is a fact that the purity of clusters is proportional to the number of clusters generated. Fig. 6 shows the numbers of clusters generated by LargeItem and PWO. it is observed that when the minimum support equals to 100% the PWO algorithm handles each transaction in an individual cluster because PWO equals to one.

To minimize the number of clusters, the merge strategy is applied with with PWO-M algorithm, and then compare the number of clusters with PWO algorithm. Fig. 7 illustrates the effect of group merging on minimizing the number of clusters depending on the minimum overlap value. Next, LargeItem, PWO, and PWO-M are put in comparison of their final number of clusters, as illustrated in Fig. 8. There are two tests for LargeItem when (Intra=1) and when (Intra=10). It is obvious that LargeItem produces two very different results. However, PWO-M returns a reasonable number of clusters while illustrating high accuracy and best stable result.

3) Experiment 3: Comparison of Different Clustering Algorithms vs. PWO: In this experiment, the following algorithms: PWO, PWO-M, LargeItem, and CLOPE are compared using the datasets and parameters in Table V.

Fig. 9 illustrates their comparison in terms of purity of resulting clusters. It was seen that PWO and PWO-M are more accurate and more stable. Second, the output number of clusters between the four algorithms are compared, illustrated in Fig. 10. It is noticed that LargeItem returns the minimum number of clusters for all datasets, while PWO and PWO-M return a very large number of clusters for Voting and

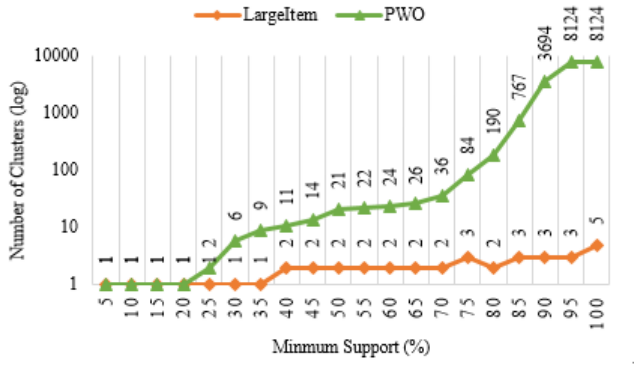

Fig. 6. Final No. of clusters using LargeItem vs. PWO on the mushroom dataset.

Fig. 7. Final No. of clusters using PWO vs. PWO-M on the mushroom dataset.

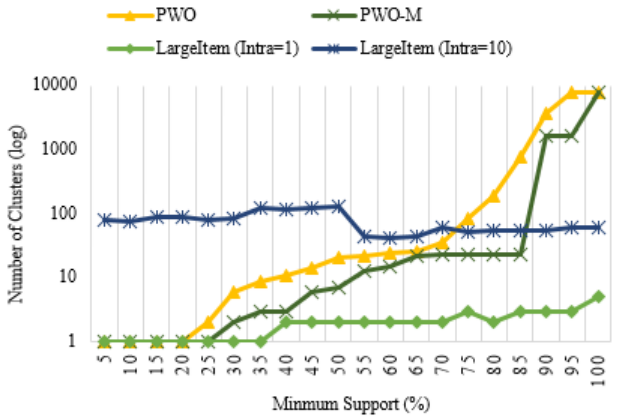

Fig. 8. Number of clusters vs. minimum support on the mushroom dataset.

Hepatitis datasets, but for the remaining datasets, they return a reasonably larger number of clusters if it is taken into consideration the gain of clusters' purity. From this test, it is concluded that PWO is measurable and its strength appears in the purity of resulting clusters.

4) Experiment 4: Evaluation of PWO Algorithm using Fixed Number Clustering : As previously explained, PWO-M is adapted to work with the case when there is a pre-defined

TABLE V. DATASETS AND SELECTED PARAMETERS

Dataset	Size	Overlap/Minimum Support
Mushroom	8124	0.7
Voting	435	0.8
Hepatitis	155	0.5
Zoo	101	0.75
Lenses	24	0.6
Adult-Stretch	20	0.6

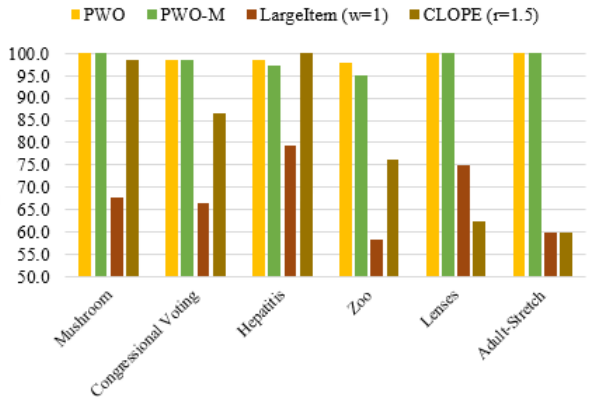

Fig. 9. Data purity in all algorithms using different datasets.

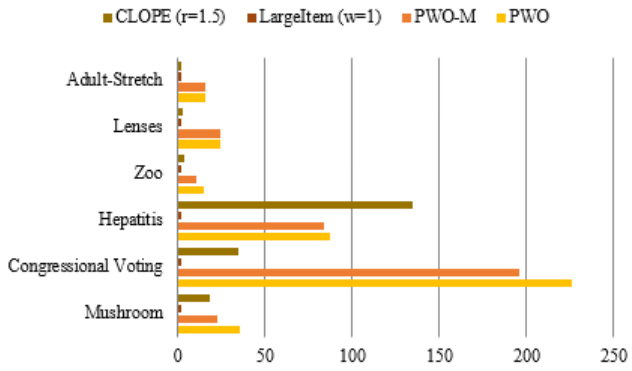

Fig. 10. Number of clusters resulting from algorithms using different datasets.

TABLE VI. SUMMARY OF FIXED-N CLUSTERING RESULTS

Dataset	No. of Classes	Purity	Avg. Classes Coverage
Mushroom	2	83.26	88.20
Zoo	7	88.12	74.28
Hepatitis	2	83.87	63.24
Voting	2	89.20	90.65

number of clusters. The result of fixed-N clustering of the Zoo dataset is 88.12% purity. The Mushroom dataset output classes having an average of 83.26% purity, while the result of the Congressional vote is 89.19% of clusters' purity. Table VI is a summary of the fixed-N clustering results. One can notice that the total average of clusters' purity is above 85%, while the average of classes' coverage is above 75%.

5) Experiment 5: Evaluation of the Overlap Estimator's Precision: To evaluate the precision of the proposed overlap estimator, at first a small training sample of the mushroom dataset is used as an input to the algorithm, 800 out of 8124

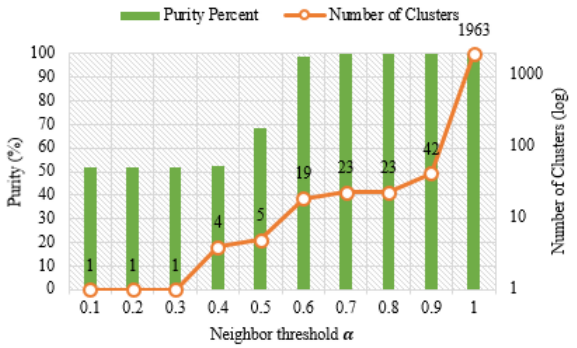

Fig. 11. Number of Clusters vs. purity in case of different neighbour's threshold values.

TABLE VII. OVERLAP THRESHOLD VALUES FOR DIFFERENT DATASET

Dataset	Total Size	Sample Size	Overlap threshold
Mushroom	8124	100	0.70
Voting	435	20	0.80
Hepatitis	155	16	0.50
Zoo	101	10	0.75
Lenses	24	12	0.60
Adult-Stretch	20	10	0.60

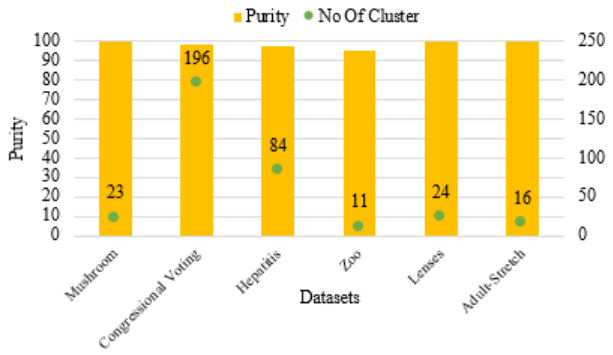

Fig. 12. Purity of clustering different datasets using the estimated overlap threshold values.

transactions selected randomly from the mushroom dataset. The overlap estimator selects a value of $\alpha = 0.7$ as an overlap threshold. Second, all the mushroom dataset is clustered and then the clusters are merged using different values of the neighbor's threshold, and the resulting number of clusters and the purity of clusters in each case are compared. The result is shown in Fig. 11, which illustrates that the value of $\alpha = 0.7$ returns the minimum number of clusters with high purity.

The same experiment is repeated using different datasets. The overlap threshold values for the datasets is listed in Table VII, while the purity and number of clusters for each dataset is displayed in Fig. 12. it is noticed that the clusters' purity for all datasets is above 95%; this indicates that the overlap estimator is very effective in detecting the behavior of data.

6) Experiment 6: Evaluation of Overlap Estimator Scalability: In this experiment, the scalability degree or the effect of the sample size is evaluated to estimate accurate neighbor's threshold. In Fig. 13, the clusters' purity is measured versus

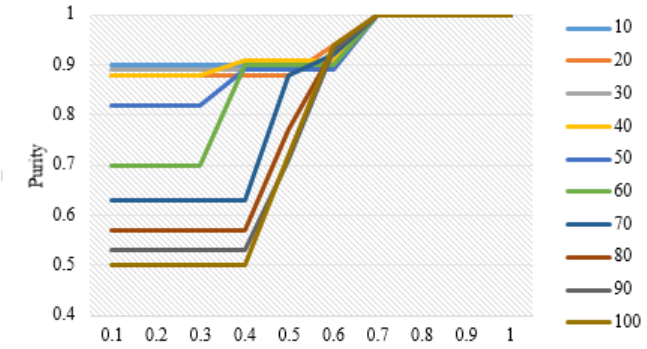

Fig. 13. Purity vs. Neighbour's threshold value using different sizes of the mushroom dataset.

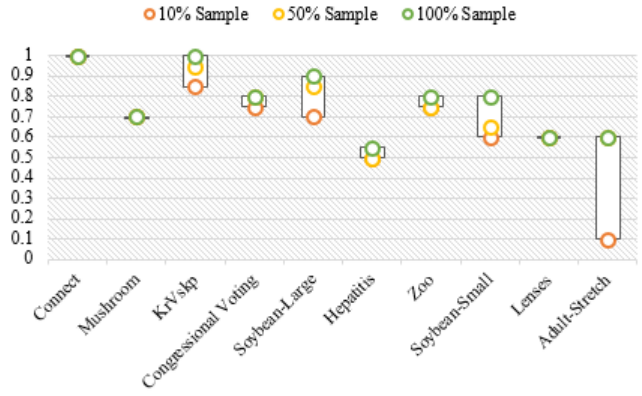

Fig. 14. Overlap Estimation Based on Sample Sizes of different datasets.

neighbor's threshold values on different percentages of the mushroom dataset size. it is noticed that all dataset sizes reach 100% purity when the neighbor's threshold value of 0.7 is used. So, the perfect estimate of overlap threshold for the mushroom dataset should be 0.7 regardless of the dataset size.

Now, the overlap estimator is applied using different sizes of dataset sample to compare the accuracy of the overlap threshold. The overlap estimator has perfectly estimated the value of the overlap threshold starting from 10% sample of the dataset, although with a 5% sample dataset (around 406 transactions) the overlap threshold is $\alpha = 0.6$ and the purity of the clusters is 93.6%, which is also acceptable.

Fig. 14 illustrates the effect of three sample sizes (10%, 50%, and 100%) on estimating the overlap threshold for the different datasets. The Adult-Stretch dataset is very small and therefore the different sample sizes produced very different estimated thresholds, i.e. it is difficult to estimate the clusters' behavior. From this experiment, it is observed that the overlap estimator could expect the best overlap threshold using a small sample if the sample is normally distributed on all classes.

Fig. 15 illustrates the clusters' purity when 10%, 50% and 100% of dataset size is used as a sample for estimation for different datasets using the PWO-M algorithm. It is seen that a nearly 100% purity is achieved from using only 10% of the dataset as a sample in the case of the Mushroom, Hepatitis, Zoo and Lenses datasets. However, this is not the case for other datasets due to the following reasons: 1) there are missing

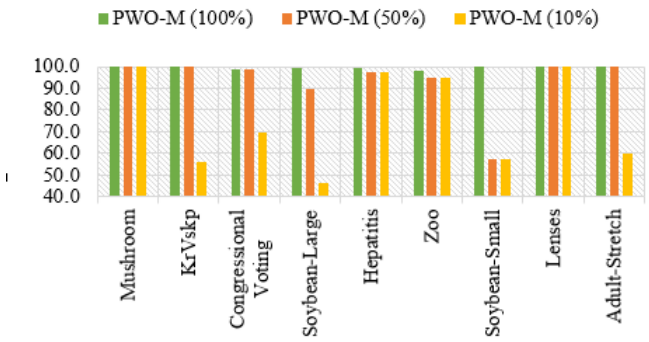

Fig. 15. Clusters' purity vs. Sample Sizes for different datasets using PWO-M.

TABLE VIII. MUSHROOM CLUSTERING PURITY VS NO. CLUSTERS

Clustering Algorithm	Purity	No. of Clusters
F-Tree ($\theta = 0.8$)	100	23
Hybrid	100	23
CLUC	100	24
CLOPE	100	30
CLICKS	100	553
Fast Clustering	99.90	23
Squeezer	99.90	24
ROCK	99.60	21
SCCADDS	99.00	19
CLICKS	97.00	19
LargeItem	95.62	14
F-Tree ($\theta = 0.3$)	95.42	16
CLICKS	87.10	14

attributes' values as in the Voting dataset. 2) The dataset could be very small as in the Adult-Stretch dataset that has 20 transactions and Soybean-Small that has 47 transactions. 3) There is a large number of clusters such as in Soybean-Large that has 19 classes. 4) There is a large number of attributes per transaction, such as in the KrVskp dataset that has 36 attributes and Soybean-Large that has 35 attributes.

Finally, three results are concluded. First, the overlap threshold value can be change for different data types. Second, the estimation of the overlap threshold using a small size of data would depend on the data type and number of clusters in the dataset. Third, the overlap estimator could improve the clusters' purity.

B. F-Tree Clustering Evaluations

1) Experiment 1: Comparing clustering algorithms without pre-defined number of clusters: In this experiment, almost all of the algorithms are compared for best purity with closer number of clusters. The algorithms are run without pre-defined number of clusters. Table VIII lists different clustering algorithms in order to evaluate the clustering purity versus the number of clustering. This table is sorted by purity in descending order and by number of clusters in ascending order. The algorithms that are given the closer number of clusters with high purity are both F-Tree [13] and Hybrid [19], which gives 100% of purity with only 23 clusters. It is also figured that LargeItem returned the minimum number of clusters but with higher purity than CLICKS [25].

TABLE IX. DIFFERENT DATASET CLUSTERING PURITIES

Algorithm	Mushroom	Car	Zoo	Hepatitis	Voting	Cancer
ROCK	77.00	77.08	-	99.35	79.00	97.20
COOLCAT	76.00	-	-	-	87.00	-
Squeezer	53.60	-	89.00	-	61.80	86.20
LIMBO	89.00	44.50	-	84.52	87.12	69.93
DELTA	89.02	30.15	-	69.67	89.43	70.97
SCCADDS	89.00	-	-	-	88.00	-
Ensemble	89.00	-	93.00	-	87.00	96.70
SF-Tree (10% seed)	99.00	89.90	93.00	79.00	87.00	96.20
CBDA	89.02	-	-	-	91.95	-
DILCA M	89.02	70.08	-	83.22	91.95	74.47
DILCA RR	89.02	70.08	-	69.67	89.43	74.47

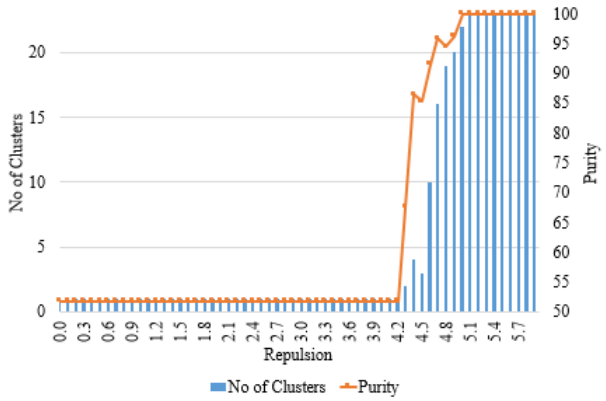

Fig. 16. Clusters' purity vs. No of clusters for different values of Repulsion of CLOPE algorithm.

2) Experiment 2: Comparing clustering algorithms with pre-defined number of clusters: In this experiment, almost all of the algorithms are compared for best purity but with pre-defined number of clusters given prior to the clustering process. In Table IX, the results are presented across different datasets. it could notice that SF-Tree is the best algorithm across different full categorical datasets such as Mushroom, Car, and Zoo. However Voting dataset is categorical but contains a lot of missing data which affects the purity of algorithm. But, in numeric dataset such as Hepatitis and Cancer, It fail to get a competitive result as this version of F-Tree is based on exact matching of feature's value or records.

3) Experiment 3: Evaluation of CLOPE cost function: It is implemented the CLOPE clustering algorithm to analyze the effect of CLOPE [24] cost function on merging pure clusters. In this experiment, there is a try to minimize the number of clusters generated from F-Tree using CLOPE cost function. On mushroom dataset, F-Tree generates 23 pure clusters. These clusters are input to CLOPE algorithm as existing clusters, then run algorithm to minimize the number of clusters. there is a try of a range of r-parameter (from 0 to 6) of CLOPE algorithm and measure the number of clusters and purity each time. In Fig. 16, it is noticed that at value of (4.9) the number of clusters is decreased, but with an effect on purity. The CLOPE cost function is failed to minimize the number of clusters in addition to the main problem of determining the best value of r-parameter or "Repulsion".

TABLE X. ANALYSIS OF SF-TREE ON DIFFERENT DATASETS

Dataset	Mushroom	Cancer	Voting	Zoo
Number of samples	100	50	25	20
DB Size	8124	699	435	101
Seed Percent(%)	1%	7%	6%	20%
Class	2	2	2	7
Entropy	1.003	0.942	1.012	2.201
CAIR	0.972	0.868	0.762	0.946
Info-Loss	0.014	0.066	0.118	0.032
E-min	0.010	0.044	0.090	0.059
Precision	0.990	0.956	0.910	0.941
Recall	0.990	0.956	0.916	0.976
F-Measure	0.990	0.956	0.911	0.940
Purity	99.00	95.60	91.00	94.10

4) Experiment 4: Analysis of Clustering with SF-Tree: The goal of this experiment is to test the dependence between the dataset and seed percentage, it is noticed that the seed can be used randomly but it is effective if the dataset size is smaller. Table X shows that in Zoo dataset it is used 20% of dataset as a seed in order to gain the purity because the Zoo is very small dataset around 101 records, while in mushroom the use of 1% of dataset is sufficient to gain a good purity value because the dataset is large enough.

X. LIMITATIONS

This clustering approach using PWO was unable to sufficiently minimize the number of resulting clusters and further research is needed to overcome this drawback. The overlap estimator framework is designed for categorical datasets; extending it to domains with continuous values will be a challenging task.

XI. CONCLUSION AND FUTURE WORK

In this paper, a new similarity measure, "PWO" is proposed to overcomes the overlapping between clusters. From the experiments, it is concluded that PWO is applicable to different categorical datasets and generates acceptable degree of clusters' purity. New clustering algorithms using PWO is presented and experiments showed that this approach is effective. PWO also can be applied in many applications, so it can be used in 1) measuring the similarity between clusters of categorical or transactional dataset, 2) measuring the best pair of clusters, and 3) classifying the dataset based on the similarity between categorical items. Since it is important to determine the best similarity threshold for different datasets, the overlap estimator framework based on the training dataset is proposed. Experiments show that only 10% of the total datasets is sufficient to detect the best similarity threshold value.

REFERENCES

[1] Z. He, X. Xu, and S. Deng, "Squeezer: an efficient algorithm for clustering categorical data," *Journal of Computer Science and Technology*, vol. 17, no. 5, pp. 611–624, 2002.

[2] Z. Huang, "A fast clustering algorithm to cluster very large categorical data sets in data mining." in *DMKD*. Citeseer, 1997.

[3] D. Gibson, J. Kleinberg, and P. Raghavan, "Clustering categorical data: An approach based on dynamical systems," *Databases*, vol. 1, 1998.

[4] H. Yan, K. Chen, and L. Liu, "Efficiently clustering transactional data with weighted coverage density," in *Proceedings of the 15th ACM international conference on Information and knowledge management*. ACM, 2006, pp. 367–376.

[5] N. Ye, *The handbook of data mining*. Lawrence Erlbaum Associates Mahwah, NJ, 2003, vol. 24.

[6] M. W. Berry and M. Browne, *Lecture notes in data mining*. World Scientific, 2006.

[7] S. Guha, R. Rastogi, and K. Shim, "Rock: A robust clustering algorithm for categorical attributes," in *Data Engineering, 1999. Proceedings., 15th International Conference on*. IEEE, 1999, pp. 512–521.

[8] H. D. Margaret, "Data mining introductory and advanced topics," *LPE: Pearson Education Publising*, 2003.

[9] S. Guha, R. Rastogi, and K. Shim, "Cure: an efficient clustering algorithm for large databases," in *ACM SIGMOD Record*, vol. 27, no. 2. ACM, 1998, pp. 73–84.

[10] D. J. Hand, H. Mannila, and P. Smyth, *Principles of data mining*. MIT press, 2001.

[11] S. Sharma, *Applied multivariate techniques*. John Wiley & Sons, Inc., 1995.

[12] K. Chen and L. Liu, ""best k": critical clustering structures in categorical datasets," *Knowledge and information systems*, vol. 20, no. 1, pp. 1–33, 2009.

[13] M. A. Mahdi, S. E. Abdel-Rahman, R. Bahgat, and I. A.Ismail, "F-tree: An algorithm for clustering transactional data using frequency tree," in *Al-Azhar University Engineering Journal, Eleventh International Conference*, vol. 5, no. 8. Al-Azhar University, Cairo, Egypt. https://arxiv.org/abs/1705.00761: JAUE, December 21 - 23 2010, pp. 101–123.

[14] C. Blake and C. J. Merz, "UCI repository of machine learning databases," 1998.

[15] D. Barbara, Y. Li, and J. Couto, "Coolcat: an entropy-based algorithm for categorical clustering," in *Proceedings of the eleventh international conference on Information and knowledge management*. ACM, 2002, pp. 582–589.

[16] P. Andritsos, P. Tsaparas, R. J. Miller, and K. C. Sevcik, "Limbo: Scalable clustering of categorical data," in *Advances in Database Technology-EDBT 2004*. Springer, 2004, pp. 123–146.

[17] L. Xia, J. Sheng-Yi, and S. Xiao-Ke, "A novel fast clustering algorithm," in *Artificial Intelligence and Computational Intelligence, 2009. AICI'09. International Conference on*, vol. 4. IEEE, 2009, pp. 284–288.

[18] J. Al-Shaqsi and W. Wang, "A clustering ensemble method for clustering mixed data," in *Neural Networks (IJCNN), The 2010 International Joint Conference on*. IEEE, 2010, pp. 1–8.

[19] S.-Y. Jiang and X. Li, "A hybrid clustering algorithm," in *Fuzzy Systems and Knowledge Discovery, 2009. FSKD'09. Sixth International Conference on*, vol. 1. IEEE, 2009, pp. 366–370.

[20] K. Wang, C. Xu, and B. Liu, "Clustering transactions using large items," in *Proceedings of the eighth international conference on Information and knowledge management*. ACM, 1999, pp. 483–490.

[21] C.-H. Yun, K.-T. Chuang, and M.-S. Chen, "An efficient clustering algorithm for market basket data based on small large ratios," in *Computer Software and Applications Conference, 2001. COMPSAC 2001. 25th Annual International*. IEEE, 2001, pp. 505–510.

[22] Y. S. Koh and R. Pears, "Transaction clustering using a seeds based approach," in *Advances in Knowledge Discovery and Data Mining*. Springer, 2008, pp. 916–922.

[23] V. Ganti, J. Gehrke, and R. Ramakrishnan, "Cactus clustering categorical data using summaries," in *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1999, pp. 73–83.

[24] Y. Yang, X. Guan, and J. You, "Clope: a fast and effective clustering algorithm for transactional data," in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2002, pp. 682–687.

[25] M. J. Zaki and M. Peters, "Clicks: Mining subspace clusters in categorical data via k-partite maximal cliques," in *Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on*. IEEE, 2005, pp. 355–356.

[26] A. Ahmad and L. Dey, "A method to compute distance between two categorical values of same attribute in unsupervised learning for categorical data set," *Pattern Recognition Letters*, vol. 28, no. 1, pp. 110–118, 2007.

[27] E. Abdu and D. Salane, "A spectral-based clustering algorithm for categorical data using data summaries," in *Proceedings of the 2nd Workshop on Data Mining using Matrices and Tensors*. ACM, 2009, p. 2.

[28] A. Nemalhabib and N. Shiri, "Cluc: a natural clustering algorithm for categorical datasets based on cohesion," in *Proceedings of the 2006 ACM symposium on Applied computing*. ACM, 2006, pp. 637–638.

[29] K. Lu, "Clustering transactions using context-based distance," no. 143, pp. 123–138, 2012.

[30] G. Ivchenko and S. Honov, "On the jaccard similarity test," *Journal of Mathematical Sciences*, vol. 88, no. 6, pp. 789–794, 1998.

[31] F. Alqadah and R. Bhatnagar, "Similarity measures in formal concept analysis," *Annals of Mathematics and Artificial Intelligence*, vol. 61, no. 3, pp. 245–256, 2011.

[32] Y. S. Koh and R. Pears, "Rare association rule mining via transaction clustering," in *Proceedings of the 7th Australasian Data Mining Conference-Volume 87*. Australian Computer Society, Inc., 2008, pp. 87–94.