Hierarchical Co-Clustering Based on Entropy Splitting

Wei Cheng [1], Xiang Zhang [2], Feng Pan [3], and Wei Wang [4]
[1]Department of Computer Science, University of North Carolina at Chapel Hill,
[2]Department of Electrical Engineering and Computer Science, Case Western Reserve University,
[3]Microsoft, [4] Department of Computer Science, University of California, Los Angeles

ABSTRACT

Two dimensional contingency tables or co-occurrence matrices arise frequently in various important applications such as text analysis and web-log mining. As a fundamental research topic, co-clustering aims to generate a meaningful partition of the contingency table to reveal hidden relationships between rows and columns. Traditional co-clustering algorithms usually produce a predefined number of flat partition of both rows and columns, which do not reveal relationship among clusters. To address this limitation, hierarchical co-clustering algorithms have attracted a lot of research interests recently. Although successful in various applications, the existing hierarchial co-clustering algorithms are usually based on certain heuristics and do not have solid theoretical background.

In this paper, we present a new co-clustering algorithm with solid information theoretic background. It simultaneously constructs a hierarchical structure of both row and column clusters which retains sufficient mutual information between rows and columns of the contingency table. An efficient and effective greedy algorithm is developed which grows a co-cluster hierarchy by successively performing row-wise or column-wise splits that lead to the maximal mutual information gain. Extensive experiments on real datasets demonstrate that our algorithm can reveal essential relationships of row (and column) clusters and has better clustering precision than existing algorithms.

Categories and Subject Descriptors

I.5.3 [**Pattern Recognition**]: Clustering-$algorithms$

General Terms

Algorithms, Experimentation, Theory

Keywords

Co-clustering, Entropy, Contingency Table, Text Analysis

1. INTRODUCTION.

Two dimensional contingency table arises frequently in various applications such as text analysis and web-log mining. Co-clustering algorithms have been developed to perform two-way clustering on both rows and columns of the contingency table. Traditional co-clustering algorithms usually generate a strict partition of the table [4, 6, 12]. This flat structure is insufficient to describe relationships between clusters. Such relationships are essential for data navigation, browsing, and so forth in many applications such as document analysis.

To combine the advantages of both co-clustering and hierarchical clustering, various hierarchical co-clustering algorithms have been recently proposed [8, 3, 15, 1, 9]. However, existing hierarchical co-clustering algorithms are usually based on certain heuristic criteria or measurements for agglomerating or dividing clusters of rows and columns. Such criteria may be domain-specific and thus suffer from lacking of generality. Another limitation of many existing hierarchical co-clustering algorithms is that they often require the number of clusters (for both rows and columns) as an input. However, accurate estimation of the number of clusters may be impossible in many applications, or require a pre-processing stage.

To overcome these limitations, we propose a hierarchical co-clustering algorithm with solid information theoretic background. Our approach aims to generate the simplest co-cluster hierarchy that retains sufficient mutual information between rows and columns in the contingency table. More specifically, the mutual information between resulting row clusters and column clusters should not differ from the mutual information between the original rows and columns by more than a small fraction (specified by the user). Finding the optimal solution for this criterion however would take exponential time. Thus, we devise an efficient greedy algorithm that grows a co-cluster hierarchy by successively performing row-wise or column-wise splits that lead to the maximal mutual information gain at each step. This procedure starts with a single row cluster and a single column cluster and terminates when the mutual information reaches a threshold. Other termination criteria (such as the desired number of row/column clusters) can be easily incorporated.

In summary, our hierarchical co-clustering algorithm has the three advantages: (1). Our algorithm builds cluster hierarchies on both rows and columns simultaneously. The relationships between clusters are explicitly revealed by the hierarchies. And the hierarchical structures inferred by our approach are useful for indexing and visualizing data, exploring the parent-child relationships, and deriving generation/specialization concepts. (2). Our algorithm uses an uniform framework to model the hierarchical co-clustering problem, and the optimality of splitting the clusters is guaranteed by rigorous proofs. (3). Our algorithm does not necessarily require prior knowledge of the number of row and column clusters. Instead, it uses a single input, the minimum percentage of mutual information retained, and automatically derives a co-cluster hierar-

CIKM'12, October 29–November 2, 2012, Maui, HI, USA.

chy. Moreover, it is also flexible to incorporate optional constraints such as the desired number of clusters.

Experiments on real datasets show that our hierarchical co-clustering algorithm performs better than many existing co-clustering algorithms.

2. RELATED WORK.

Co-clustering is a branch of clustering methods that clusters both rows and columns simultaneously. The problem of co-clustering has been studied extensively in recent literatures. These clustering algorithms generate *flat* partitions of rows and columns. However, a taxonomy structure can be more beneficial than a flat partition for many applications such as document clustering. In this section, we present a survey of recent co-clustering algorithms.

A pioneering co-clustering algorithm based on information theory was proposed by Dhillon *et.al.* in [6]. Taking the numbers of row and column clusters as input, the algorithm generate a flat partition of data matrix into row clusters and column clusters which maximizes the mutual information between row and column clusters. The idea is generalized into a meta algorithm in [2]. It is proven in [2] that besides mutual information, any Bregman divergence can be used in the objective function and the two-step iteration algorithm can always find a co-cluster by converging the objective function to a local minimum. Even though the co-clustering model proposed in these papers is also rooted in information theory, such as mutual information and relative entropy, our approach generates hierarchical cluster structures. This key difference entails different objective functions and, more importantly, different optimization techniques. In addition, linear algebra methods are also applied to derive co-clusters[4, 12].

By integrating hierarchical clustering and co-clustering, hierarchical co-clustering aims at simultaneously constructing hierarchical structures for two or more data types. Hierarchical co-clustering has recently received special attentions [8, 3]. A hierarchical divisive co-clustering algorithm is proposed in [15] to simultaneously find document clusters and the associated word clusters. It has also been incorporated into a novel artist similarity quantifying framework for the purpose of assisting artist similarity quantification by utilizing the style and mood clusters information [1]. Both hierarchical agglomerative and divisive co-clustering methods have been applied to organize the music data [9].

3. PRELIMINARY.

We denote the two-dimensional contingency table as T. $R = \{r_1, r_2, ..., r_n\}$ represents the set of rows of T, where r_i is the i^{th} row. $C = \{c_1, c_2, ..., c_m\}$ represents the set of columns, where c_j is the j^{th} column. The element at the i^{th} row and j^{th} column is denoted by T_{ij}. For instance, in a word-document table, each document is represented by a row and each word maps to a column. Each element stores the frequency of a word in a document.

We can compute a joint probability distribution by *normalizing* elements in the table. Let X and Y be two discrete random variables that take values in R and C respectively. The normalized table can be considered as a joint probability distribution of X and Y. We denote $p(X = r_i, Y = c_j)$ by $p(r_i, c_j)$ for convenience in the remainder of this paper.

A *co-cluster* consists of a set of row clusters and a set of column clusters. We denote the set of row clusters as $\hat{R}$,

$$\hat{R} = \{\hat{r}_1, \hat{r}_2, ..., \hat{r}_l | \hat{r}_i \subseteq R, \hat{r}_i \cap \hat{r}_j = \emptyset, i \neq j\}$$

where $\hat{r}_i$ represents the i^{th} row cluster.

Similarly, we denote the set of column clusters as $\hat{C}$,

$$\hat{C} = \{\hat{c}_1, \hat{c}_2, ..., \hat{c}_k | \hat{c}_j \subseteq C, \hat{c}_j \cap \hat{c}_i = \emptyset, j \neq i\}$$

where $\hat{c}_j$ represents the j^{th} row cluster. We denote the number of clusters in $\hat{R}$ as $L_{\hat{R}} = |\hat{R}|$, and the number of clusters in $\hat{C}$ as $L_{\hat{C}} = |\hat{C}|$. Given the sets of row and column clusters, a co-cluster can be considered as a "reduce" table $\hat{T}$ from T. Each row (column) in $\hat{T}$ represents a row (column) cluster. Each element in $\hat{T}$ is the aggregation of the corresponding elements in T,

$$\hat{T}_{ij} = \sum \{T_{uv} | r_u \in \hat{r}_i, c_u \in \hat{c}_j\}$$

Let $\hat{X}$ and $\hat{Y}$ be two discrete random variables that take values in $\hat{R}$ and $\hat{C}$ respectively. A normalized reduced table can be considered as a joint probability distribution of $\hat{X}$ and $\hat{Y}$. We will denote $p(\hat{X} = \hat{r}_i, \hat{Y} = \hat{c}_j)$ by $p(\hat{r}_i, \hat{c}_j)$ for convenience.

Note that the original contingency table can be viewed as a co-cluster by regarding each single row (column) as a row (column) cluster. Given any co-cluster $(\hat{R}, \hat{C})$ on a contingency table, we employ the mutual information between $\hat{X}$ and $\hat{Y}$ to measure the relationship between row clusters and column clusters.

$$I(\hat{X}, \hat{Y}) = \sum_{\hat{r} \in \hat{R}} \sum_{\hat{c} \in \hat{C}} p(\hat{r}, \hat{c}) log_2 \frac{p(\hat{r}, \hat{c})}{p(\hat{r}) p(\hat{c})}$$

As you may observe, the mutual information of the original table $I(X, Y)$ is larger than the mutual information of the aggregated table $I(\hat{X}, \hat{Y})$, due to clustering. This is in fact a property held by co-clustering described in Theorem 3.1.

In order to prove Theorem 3.1, we first prove the following lemmas based on the theorems proven by Dhillon *et.al.* [6].

LEMMA 3.1. *Given two co-clusters,* $\{\hat{R}^{(1)}, \hat{C}^{(1)}\}$ *and* $\{\hat{R}^{(2)}, \hat{C}^{(1)}\}$, *where* $\hat{R}^{(2)}$ *is generated by splitting a row cluster in* $\hat{R}^{(1)}$. *Then*

$$I(\hat{X}^{(1)}, \hat{Y}^{(1)}) \leq I(\hat{X}^{(2)}, \hat{Y}^{(1)})$$

Proof: Assume that $\hat{R}^{(2)}$ is generated by splitting $\hat{r}_1^{(1)} \in \hat{R}^{(1)}$ into $\hat{r}_1^{(2)}$ and $\hat{r}_2^{(2)}$. We have

$$\begin{aligned} & I(\hat{X}^{(2)}, \hat{Y}^{(1)}) - I(\hat{X}^{(1)}, \hat{Y}^{(1)}) \\ &= H(\hat{Y}^{(1)} | \hat{X}^{(1)}) - H(\hat{Y}^{(1)} | \hat{X}^{(2)}) \\ &= - \sum_{\hat{c}^{(1)} \in \hat{Y}^{(1)}} p(\hat{r}_1^{(1)}, \hat{c}^{(1)}) log p(\hat{c}^{(1)} | \hat{r}_1^{(1)}) + \sum_{\hat{c}^{(1)} \in \hat{Y}^{(1)}} p(\hat{r}_1^{(2)}, \hat{c}^{(1)}) log p(\hat{c}^{(1)} | \hat{r}_1^{(2)}) \\ &\quad + \sum_{\hat{c}^{(1)} \in \hat{Y}^{(1)}} p(\hat{r}_2^{(2)}, \hat{c}^{(1)}) log p(\hat{c}^{(1)} | \hat{r}_2^{(2)}) \end{aligned}$$

Because $\hat{r}_1^{(2)} \cup \hat{r}_2^{(2)} = \hat{r}_1^{(1)}$, we have

$$p(\hat{r}_1^{(1)}, \hat{c}^{(1)}) = p(\hat{r}_1^{(2)}, \hat{c}^{(1)}) + p(\hat{r}_2^{(2)}, \hat{c}^{(1)}), \forall \hat{c}^{(1)} \in \hat{Y}^{(1)}$$

Therefore,

$$\begin{aligned} & I(\hat{X}^{(2)}, \hat{Y}^{(1)}) - I(\hat{X}^{(1)}, \hat{Y}^{(1)}) \\ &= \sum_{\hat{c}^{(1)} \in \hat{Y}^{(1)}} p(\hat{r}_1^{(2)}, \hat{c}^{(1)}) log \frac{p(\hat{c}^{(1)} | \hat{r}_1^{(2)})}{p(\hat{c}^{(1)} | \hat{r}_1^{(1)})} + \sum_{\hat{c}^{(1)} \in \hat{Y}^{(1)}} p(\hat{r}_2^{(2)}, \hat{c}^{(1)}) log \frac{p(\hat{c}^{(1)} | \hat{r}_2^{(2)})}{p(\hat{c}^{(1)} | \hat{r}_1^{(1)})} \\ &= p(\hat{r}_1^{(2)}) D(p(\hat{c}^{(1)} | \hat{r}_1^{(2)}) || p(\hat{c}^{(1)} | \hat{r}_1^{(1)})) + p(\hat{r}_2^{(2)}) D(p(\hat{c}^{(1)} | \hat{r}_2^{(2)}) || p(\hat{c}^{(1)} | \hat{r}_1^{(1)})) \end{aligned}$$

where $D(p(\hat{r}_1^{(2)}, \hat{c}^{(1)}) || p(\hat{r}_1^{(1)}, \hat{c}^{(1)}))$ is the relative entropy (KL-divergence) between $p(\hat{c}^{(1)} | \hat{r}_1^{(2)})$ and $p(\hat{c}^{(1)} | \hat{r}_1^{(1)})$, which is always non-negative (by definition). Therefore

$$I(\hat{X}^{(2)}, \hat{Y}^{(1)}) - I(\hat{X}^{(1)}, \hat{Y}^{(1)}) \geq 0$$

and then

$$I(\hat{X}^{(1)}, \hat{Y}^{(1)}) \leq I(\hat{X}^{(2)}, \hat{Y}^{(1)})$$

□

Similarly, we have

LEMMA 3.2. *Given two co-clusters, $\{\hat{R}^{(1)}, \hat{C}^{(1)}\}$ and $\{\hat{R}^{(1)}, \hat{C}^{(2)}\}$, and $\hat{C}^{(2)}$ is generated by splitting one column cluster in $\hat{C}^{(1)}$. Then*

$$I(\hat{X}^{(1)}, \hat{Y}^{(1)}) \leq I(\hat{X}^{(1)}, \hat{Y}^{(2)})$$

The above two lemmas state that splitting either row or column-wise clusters increases the mutual information between the two sets of clusters. Hence, we can obtain the original contingency table (i.e., each row/column itself is a row/column cluster) by performing a sequence of row-wise splits or column-wise splits on a co-cluster. By Lemmas 3.1 and 3.2, the mutual information monotonically increases after each split, which leads to the following theorem.

THEOREM 3.1. *The mutual information of a co-clustering, $I(\hat{X}, \hat{Y})$, always increases when any one of its row or column clusters is split, until the mutual information reaches its maximal value, $I(X, Y)$, where each row and column is considered as a single cluster.*

The monotonicity property of mutual information leads to the following problem definition.

Problem Definition.

Given a normalized two-dimensional contingency table, T, and a threshold $\theta (0 < \theta < 1)$, find a hierarchical co-clustering containing a minimum number of the leaf row clusters $\hat{R}$ and leaf column clusters $\hat{C}$, such that the mutual information corresponding to co-clustering $\{\hat{R}, \hat{C}\}$ satisfies $\frac{I(\hat{X},\hat{Y})}{I(X,Y)} \geq \theta$. Optionally, a user can specify desired number of row or column clusters ($L_{\hat{R}} = max_r$ or $L_{\hat{C}} = max_c$) and ask for a co-cluster with maximal mutual information.

4. CO-CLUSTERING ALGORITHM.

In this section, we present the details of our co-clustering algorithm. The monotonicity property of mutual information stated in Lemmas 3.1 and 3.2 inspires us to develop a greedy divisive algorithm that optimizes the objective function $I(\hat{X}, \hat{Y})$ at each step. The main routine is presented in Section 4.1.

4.1 The Main Algorithm.

The pseudocode of the algorithm is shown in Figure 1. In Step 1 of the main function *Co-Clustering()*, function *InitialSplit()* is called to generate the initial co-cluster $\{\hat{R}^{(0)}, \hat{C}^{(0)}\}$ with two row clusters and two column clusters. In Step 2, the joint distribution $p(\hat{X}, \hat{Y})$ of this initial co-cluster is calculated. Then the algorithm goes through iterations. During each iteration, a split is performed to maximize the mutual information of the co-cluster. In Steps 5 and 6, each row or column cluster s_i is examined by function *SplitCluster()* to determine the highest gain in mutual information, $\delta I_i^{(k)}$, which can be brought by an optimal split on s_i. (s_{i1} and s_{i2} denote the resulting clusters after split.) Steps 7 to 9 select the row or column cluster whose split gives the highest gain $\delta I_i^{(k)}$, and perform the split. In Step 10, the joint distribution $p(\hat{X}, \hat{Y})$ is updated according to the new co-cluster $\{\hat{R}^{(k+1)}, \hat{C}^{(k+1)}\}$. The algorithm continues until the mutual information ratio $\frac{I(\hat{X},\hat{Y})}{I(X,Y)}$ reaches the threshold, θ, and/or the number of clusters (row or column) reaches the number of desired clusters, denoted by max_c and max_r. Note that the termination condition can be easily modified to suit users' needs.

4.2 Initial Split.

Function *InitialSplit()* splits the contingency table into two row clusters and two column clusters. In Step 1, the joint distribution is set to the normalized table T. In Step 2, all rows are considered as in a single row cluster s_1 and all columns are considered as in a single column cluster s_2. They are then split in Steps 3 and 4 by calling the function *SplitCluster()*. The initial co-cluster $\{\hat{R}^{(0)}, \hat{C}^{(0)}\}$ and the corresponding mutual information $I^{(0)} = I(\hat{X}^{(0)}, \hat{Y}^{(0)})$ are calculated accordingly in Steps 5 and 6.

Note that we split both row clusters and column clusters in this initial step. To ensure a good initial split, when the function *SplitCluster()* is called, we tentatively treat each row as an individual cluster so that the initial column clusters are created by taking into account the row distribution. By the same token, we also tentatively treat each column as an individual cluster when we create the initial row clusters.

main function *Co-Clustering()*
Input:
- Normalized table, T
- Minimal threshold of $\frac{I(\hat{X},\hat{Y})}{I(X,Y)}$, θ

Optional Input:
- Maximal number of row clusters, max_r
- Maximal number of column clusters, max_c

Output: Co-cluster $\{\hat{R}, \hat{C}\}$
Method:
1. $\{\hat{R}^{(0)}, \hat{C}^{(0)}, I^{(0)}\} = InitialSplit(T)$
2. calculate distribution $p(\hat{X}, \hat{Y})$ according to $\{\hat{R}^{(0)}, \hat{C}^{(0)}\}$
3. k=0
4. Do
5. for each cluster $s_i, s_i \in \hat{R}^{(k)} \bigcup \hat{C}^{(k)}$
6. $\{s_{i1}, s_{i2}, \delta I_i^{(k)}\} = SplitCluster(s_i, p(\hat{X}, \hat{Y}))$
7. let $\delta I_j^{(k)} = max\{\delta I_i^{(k)}\}$
8. $I^{(k+1)} = I^{(k)} + \delta I_j^{(k)}$
9. $\{\hat{R}^{(k+1)}, \hat{C}^{(k+1)}\} = (\hat{R}^{(k)} \bigcup \hat{C}^{(k)} - s_j) \bigcup \{s_{j1}, s_{j2}\}$
10. update $p(\hat{X}, \hat{Y})$ according to $\{\hat{R}^{(k+1)}, \hat{C}^{(k+1)}\}$
11. $k = k + 1$
12. While $\frac{I(\hat{X},\hat{Y})}{I(X,Y)} < \theta$ and/or $|\hat{R}^{(k)}| < max_r$ and/or $|\hat{C}^{(k)}| < max_c$
13. return $\{\hat{R}^{(k)}, \hat{C}^{(k)}\}$

function *InitialSplit*(T)
Input: Normalized table T
Output: $\{\hat{R}^{(0)}, \hat{C}^{(0)}, I^{(0)}\}$
Method:
1. $p(X, Y) = T$
2. $s_1 = R, s_2 = C$
3. $\{s_{11}, s_{12}, \delta I_1\} = SplitCluster(s_1, p(X, Y))$
4. $\{s_{21}, s_{22}, \delta I_2\} = SplitCluster(s_2, p(X, Y))$
5. $\hat{R}^{(0)} = \{s_{11}, s_{12}\}, \hat{C}^{(0)} = \{s_{21}, s_{22}\}$
6. $I^{(0)} = I(\hat{X}^{(0)}, \hat{Y}^{(0)})$
7. return $\{\hat{R}^{(0)}, \hat{C}^{(0)}, I^{(0)}\}$

Figure 1: Algorithm

4.3 Cluster Splitting.

According to Lemmas 3.1 and 3.2, a split will never cause $I(\hat{X}, \hat{Y})$ decrease. In addition, only the split cluster may contribute to the increase of $I(\hat{X}, \hat{Y})$. Suppose that a row cluster $\hat{r}^{(1)}$ is split into $\hat{r}_1^{(2)}$ and $\hat{r}_2^{(2)}$, the increase in $I(\hat{X}, \hat{Y})$ is

$$\delta I = I(\hat{X}^{(2)}, \hat{Y}^{(1)}) - I(\hat{X}^{(1)}, \hat{Y}^{(1)})$$
$$= p(\hat{r}_1^{(2)}) D(p(\hat{c}^{(1)}|\hat{r}_1^{(2)}) || p(\hat{c}^{(1)}|\hat{r}_1^{(1)})) + p(\hat{r}_2^{(2)}) D(p(\hat{c}^{(1)}|\hat{r}_2^{(2)}) || p(\hat{c}^{(1)}|\hat{r}_1^{(1)})) \quad (1)$$

Therefore, *SplitCluster()* can calculate the maximal value of δI by examining each cluster to be split separately. However, it may still take exponential time (with respect to the cluster size) to find the optimal split. Therefore, *SplitCluster()* adopts a greedy algorithm that can effectively produce a good split achieving a local maximum in δI. Elements in the cluster are initially randomly grouped into two sub-clusters. A sequence of iterations are tak-

function ***SplitCluster***$(s, p(\hat{X}, \hat{Y}))$
Input: Cluster s, $s \in \hat{R} \bigcup \hat{C}$
- Current joint distribution $p(\hat{X}, \hat{Y})$

Output:
- two sub-clusters of s, s_1 and s_2, s.t. $s_1 \bigcup s_2 = s$, $s_1 \bigcap s_2 = \emptyset$
- δI, the increase in $I(\hat{X}, \hat{Y})$ achieved by splitting s

Method:
1. if s is a column cluster, $p(\hat{X}, \hat{Y}) = p(\hat{X}, \hat{Y})^T$
2. randomly split s into two clusters, s_1 and s_2.
3. calculate $p(\hat{Y}|s_1)$, $p(\hat{Y}|s_2)$ and δI accordingly
4. Do
5. for each element x_i in s.
6. assign x_i to cluster s', where

$$s' = argmin_{j=1,2} D(p(\hat{Y}|x_i)||p(\hat{Y}|s_j))$$

7. update $p(\hat{Y}|s_1)$, $p(\hat{Y}|s_2)$ and δI accordingly
8. Until δI converges
9. return s_1, s_2 and δI

Figure 2: Function *SplitCluster()*

en to re-assign each element to its closer sub-cluster according to KL-divergence until δI converges.

The details of function *SplitCluster*() are shown in Figure 2. In Step 1, the joint probability distribution $p(\hat{X}, \hat{Y})$ is transposed if the input cluster s is a column cluster so that column clusters can be split in the same way as row clusters. In Step 2, cluster s is randomly split into two clusters. In Step 3, δI is calculated according to Equation 1, and the weighted mean conditional distributions of $\hat{Y}$ for both clusters s_1 and s_2 ($p(\hat{Y}|s_1)$ and $p(\hat{Y}|s_2)$) are calculated according to Equation 2.

$$p(\hat{X} = s_i) = \sum_{x_j \in s_i} p(X = x_j)$$

$$p(\hat{Y}|s_i) = \sum_{x_j \in s_i} \frac{p(X = x_j)}{p(\hat{X} = s_i)} * p(\hat{Y}|x_j) \quad (2)$$

From Step 5 to Step 7, each element x_i in cluster s is re-assigned to the cluster (s_1 or s_2) which can minimize the KL-Divergence between $p(\hat{Y}|x_i)$ and $p(\hat{Y}|s_j)$. $p(\hat{Y}|s_1)$, $p(\hat{Y}|s_2)$ and δI are updated at the end of each iteration. The procedure repeats until δI converges. In Step 9, the two sub-clusters s_1 and s_2, and δI are returned. In order to prove that function *SplitCluster*() can find a split that achieves local maximum in δI, we need to prove that the re-assignment of element x_i in Steps 4-8 can monotonically increase δI. Since the same principle is used to split row clusters and column clusters, without loss of generality, we only prove the case of splitting row clusters.

A similar cluster split algorithm was used in [5] which re-assigns elements among k clusters. It is proven that such re-assignment can monotonically decrease the sum of within-cluster JS-divergence of all clusters which is

$$Q(\{s_1, s_2, ..s_k\}) = \sum_{i=1}^{k} \sum_{x_j \in s_i} p(X = x_j) * D(p(\hat{Y}|x_j)||p(\hat{Y}|s_i))$$

In our function *SplitCluster()*, we only need to split the cluster into two sub-clusters. Therefore, we show the proof for a special case where $k = 2$. The following lemma was proven in [5].

LEMMA 4.1. *Given cluster s containing n elements ($(\hat{Y}|x_i)$), the weighted mean distribution of the cluster ($(\hat{Y}|s)$) has the lowest weighted sum of KL-divergence of $p(\hat{Y}|s)$ and $p(\hat{Y}|x_i)$. That is,$\forall q(\hat{Y})$, we have*

$$\sum_{i=1}^{n} p(x_i) \cdot D(p(\hat{Y}|x_i)||q(\hat{Y})) \geq \sum_{i=1}^{n} p(x_i) \cdot D(p(\hat{Y}|x_i)||p(\hat{Y}|s))$$

THEOREM 4.1. *When splitting cluster s into two subclusters, s_1 and s_2, the re-assignment of elements in s as shown in Steps 5-7 of function SplitCluster() can monotonically decrease the sum of within-cluster JS-divergence of the two sub-clusters, s_1 and s_2.*

Proof: Let $Q_l\{s_1, s_2\}$ and $Q_{l+1}\{s_1, s_2\}$ be the sum of within-cluster JS-divergence of the two clusters before and after the l^{th} re-assignment of elements, respectively. And let $p_l(\hat{Y}|s_i)$ and $p_{l+1}(\hat{Y}|s_i)$ be the corresponding weighted mean conditional distributions of sub-clusters before and after the l^{th} re-assignment. We will prove that $Q_{l+1}\{s_1, s_2\} \leq Q_l\{s_1, s_2\}$. Assume that the two clusters after reassignment are s_1^* and s_2^*.

$$\begin{aligned} Q_l\{s_1, s_2\} &= \sum_{i=1}^{2} \sum_{x_j \in s_i} p(X = x_j) * D(p(\hat{Y}|x_j)||p_l(\hat{Y}|s_i)) \\ &\geq \sum_{i=1}^{2} \sum_{x_j \in s_i} p(X = x_j) * D(p(\hat{Y}|x_j)||p_l(\hat{Y}|s_i^*)) \\ &= \sum_{i=1}^{2} \sum_{x_j \in s_i^*} p(X = x_j) * D(p(\hat{Y}|x_j)||p_l(\hat{Y}|s_i^*)) \\ &\geq \sum_{i=1}^{2} \sum_{x_j \in s_i^*} p(X = x_j) * D(p(\hat{Y}|x_j)||p_{l+1}(\hat{Y}|s_i^*)) \\ &= Q_{l+1}\{s_1, s_2\} \end{aligned}$$

The first inequality is a result of Step 6 in *SplitCluster()* and the second inequality is due to Step 7 in *SplitCluster()* and Lemma 4.1. Therefore, we prove that the re-assignment of elements in s can monotonically decrease $Q(\{s_1, s_2\})$. □

Note that the sum of δI and $Q(\{s_1, s_2\})$ is a constant [5], which is shown in Equation 3.

$$\begin{aligned} &\sum_{x_j \in s} p(x_j) D(p(\hat{Y}|x_j)||p(\hat{Y}|s)) \quad (3) \\ &= \sum_{i=1}^{2} \sum_{x_j \in s_i} p(x_j) D(p(\hat{Y}|x_j)||p(\hat{Y}|s_i)) + \sum_{i=1}^{2} p(s_1) D(p(\hat{Y}|s_i)||p(\hat{Y}|s)) \\ &= Q(\{s_1, s_2\}) + \delta I \end{aligned}$$

Therefore, since the re-assignment process monotonically decreases $Q(\{s_1, s_2\})$, it will monotonically increase δI as a result. Thus function *SplitCluster*() can find a split that achieves local maximum in δI.

5. EXPERIMENTS.

In this section, we perform extensive experiments on real data to evaluate the effectiveness of our co-clustering algorithm. In Sections 5.2, we use real datasets for experiments. We compare the quality of the clusters generated by our method with those generated by previous co-clustering algorithms. We use *micro-averaged precision* [6, 12] as the quality measurement.

5.1 Experimental Settings on Real Dataset.

In this section, we describe the experimental settings. We use 20 Newsgroup dataset from UCI [1]. We preprocess the 20 Newsgroup dataset to build the corresponding two dimensional contingency table. Each document is represented by a row in the table and 2000 distinct words are selected to form 2000 columns. Words are selected using the same method as in [14]. In order to compare the quality of clusters generated by our method with those of previous algorithms, we generate several subsets of the 20 Newsgroup dataset using the method in [12, 14, 6]. Each subset consists of several major newsgroups and a subset of the documents in each selected newsgroups. The details are listed in Table 1. As in [12, 14, 6], each of these subsets has two versions, one includes the subject lines of all documents and the other does not. We use $dataset_{subject}$ and $dataset$ to denote these two versions respectively.

[1] http://kdd.ics.uci.edu/databases/20newsgroups/20newsgroups.html

Table 1: Subsets of 20 Newsgroups used in Section 5.2

Dataset	Newsgroups	# documents per group	Total # documents
Multi5	comp.graphics, rec.motorcycles, rec.sports.baseball, sci.space, talk.politics.mideast	100	500
Multi10	alt.atheism, comp.sys.mac.hardware, misc.forsale, rec.autos, rec.sport.hockey sci.crypt, sci.electronics, sci.med, sci.space, talk.politics.guns	50	500

Table 2: micro-averaged precision on subsets of 20 Newsgroup

Mehtod	HICC		NVBD		ICC		HCC	
	m-pre	# clusters	m-pre	# clusters	m-pre	# clusters	m-pre	# clusters
Multi5$_{subject}$	**0.96**	30	0.93	5	0.89	5	0.72	5
Multi5	**0.96**	30	N/A		0.87	5	0.71	5
Multi10$_{subject}$	**0.74**	60	0.67	10	0.54	10	0.44	10
Multi10	**0.74**	60	N/A		0.56	10	0.61	10

Mehtod	Single-Link		Complete-Link		UPGMA		WPGMA	
	m-pre	# clusters	m-pre	# clusters	m-pre	# clusters	m-pre	# clusters
Multi5$_{subject}$	0.27	30	0.89	30	0.73	30	0.65	30
Multi5	0.29	30	0.85	30	0.59	30	0.71	30
Multi10$_{subject}$	0.24	60	0.67	60	0.60	60	0.58	60
Multi10	0.24	60	0.60	60	0.61	60	0.62	60

5.2 Comparison with Previous Algorithms.

In this section, we compare our co-clustering algorithm with several previous algorithms. For all the datasets, we empirically set $\theta = 0.7$ in our algorithm.

The state of the art co-clustering algorithms used for comparison are: (1). NBVD [12]: Co-clustering by block value decomposition. This algorithm solves the co-clustering problem by matrix decomposition. (2). ICC [6]: Information-theoretic co-clustering. This algorithm is also based on information theoretic measurements and considers the contingency table as a joint probability distribution. (3). HCC [11]: a hierarchical co-clustering algorithm. HCC brings together two interrelated but distinct themes from clustering: hierarchical clustering and co-clustering. (4). Linkage [7]: a set of agglomerative hierarchical clustering algorithms based on linkage metrics. Four different linkage metrics were used in our experiments, i.e., *Single-Link*, *Complete-link*, *UPGMA* (average), *WPGMA* (weighted average).

There are other existing co-clustering/clustering algorithms, such as [14, 10, 13], which conducted experiments on the same subsets in Table 1. Since NVBD and ICC outperform these algorithms in terms of micro-averaged precision, we will not furnish a direct comparison with them. For convenience, we use HICC to represent our algorithm and use m-pre to represent micro-averaged precision. For the number of word clusters, ICC generates about $60 - 100$ word clusters as reported in [6] while our algorithm HICC generates about $50 - 80$ word clusters. The number of word clusters generated by NVBD is not reported in [12]. While in the Linkage algorithms, since they only cluster the rows, each column can be considered as a column cluster. The comparison of micro-averaged precision on all datasets in Table 1 is shown in Table 2. In Table 2, the micro-averaged precision decrease slightly after we merge our original clusters into the same number of clusters as NVBD and ICC. This is because cluster merge may over-penalize the incorrectly labelled documents. Nevertheless, our algorithm is still the winner in all cases. The Single-linkage metric has a very low precision comparing with all other algorithms. The reason may be that using the shortest distance between two clusters as the inter-cluster distance suffers from the high dimensionality and the noise in the dataset.

6. CONCLUSIONS.

In this paper, we present a hierarchical co-clustering algorithm based on entropy splitting to analyze two-dimensional contingency tables. Taking advantage of the monotonicity of the mutual information of co-cluster, our algorithm uses a greedy approach to look for simplest co-cluster hierarchy that retains sufficient mutual information in the original contingency table. The cluster hierarchy captures rich information on relationships between clusters and relationships between elements in one cluster. Extensive experiments demonstrate that our algorithm can generate clusters with better precision quality than previous algorithms and can effectively reveal hidden cluster structures.

7. REFERENCES

[1] T. L. B. Shao and M. Ogihara. Quantify music artist similarity based on style and mood. In *WIDM '08*, pages 119–124, 2008.

[2] A. Banerjee, I. Dhillon, J. Ghosh, S. Merugu, and D. S. Modha. A generalized maximum entropy approach to bregman co-clustering and matrix approximation. In *SIGKDD'04 Conference Proceedings*, 2004.

[3] R. P. D. Ienco and R. Meo. Parameter-free hierarchical co-clustering by n-ary splits. In *Machine Learning and Knowledge Discovery in Databases*, pages 580–595, 2009.

[4] I. S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitionning. In *SIGKDD '01 Conference Proceedings*, 2001.

[5] I. S. Dhillon, S. Mallela, and R. Kumar. A divisive information-theoretic feature clustering algorithm for text classification. *Journal of Machine Learning Research*, 3:1265–1287, 2003.

[6] I. S. Dhillon, S. Mallela, and D. S. Modha. Information-theoretic co-clustering. In *SIGKDD '03 Conference Proceedings*, 2003.

[7] M. F. A survery of recent advances in hierarchical clustering algorithms. *Computer Journal*, 26(4):354–359, 1983.

[8] M. Hosseini and H. Abolhassani. Hierarchical co-clustering for web queries and selected urls. In *Web Information Systems Engineering-WISE 2007*, pages 653–662, 2007.

[9] T. L. Jingxuan Li, Bo Shao and M. Ogihara. Hierarchical co-clustering: a new way to organize the music data. In *IEEE Transactions on Multimedia*, pages 1–25, 2011.

[10] D. Lee and H.S.Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791, 1999.

[11] J. Li and T. Li. Hcc: a hierarchical co-clustering algorithm. In *SIGIR'10*, pages 861–862, 2010.

[12] B. Long, Z. Zhang, and P. S. YU. Co-clustering by block value decomposition. In *SIGKDD'05 Conference Proceedings*, 2005.

[13] R.El-Yaniv and O.Souroujon. Iterative double clustering for unsupervised and semi-supervised learning. In *ECML*, pages 121–132, 2001.

[14] N. Slonim and N. Tishby. Document clustering using word clusters via the information bottleneck method. In *ACM SIGIR*, 2000.

[15] G. Xu and W. Y. Ma. Building implicit links from content for forum search. In *SIGIR '06*, pages 300–307, 2006.