Proceedings of International Joint Conference on Neural Networks, San Jose, California, USA, July 31 – August 5, 2011

Evolutionary Spectral Co-Clustering

Nathan Green, Manjeet Rege, Xumin Liu, and Reynold Bailey
Department of Computer Science, Rochester Institute of Technology
{nsg8767, mr, xl, rjb}@cs.rit.edu

***Abstract*—Co-clustering is the problem of deriving sub-matrices from the larger data matrix by simultaneously clustering rows and columns of the data matrix. Traditional co-clustering techniques are inapplicable to problems where the relationship between the instances (rows) and features (columns) evolve over time. Not only is it important for the clustering algorithm to adapt to the recent changes in the evolving data, but it also needs to take the historical relationship between the instances and features into consideration. We present ESCC, a general framework for evolutionary spectral co-clustering. We are able to efficiently co-cluster evolving data by incorporation of historical clustering results. Under the proposed framework, we present two approaches, Respect To the Current (RTC), and Respect To Historical (RTH). The two approaches differ in the way the historical cost is computed. In RTC, the present clustering quality is of most importance and historical cost is calculated with only one previous time-step. RTH, on the other hand, attempts to keep instances and features tied to the same clusters between time-steps. Extensive experiments performed on synthetic and real world data, demonstrate the effectiveness of the approach.**

***Keywords*-data mining; clustering; co-clustering; evolving data; spectral clustering;**

I. Introduction

Clustering is the classification of data instances into different groups (clusters) such that instances in one group are similar together and dissimilar from another group. Traditional approaches deal with homogenous data instances, i.e. instances having the same data type, and are grouped together using some of the well known clustering algorithms [1]. Co-clustering on the other hand (a.k.a *bi-clustering*), involves clustering the instances and features simultaneously, and has received extensive attention recently. Typically, data is stored in a matrix W where rows and columns denote the instances and features, respectively. An entry in the matrix W_{ij} signifies the level of association between the instance i and the feature j. Co-clustering leads to deriving submatrices of this data matrix. In [2], co-clustering is defined by a pair of maps from rows to row-clusters and from columns to column-clusters inducing clustered random variables. Optimal co-clustering is then derived based on the one that leads to the largest mutual information between the clustered random variables. [3] have applied this algorithm to co-cluster auditory scenes and audio elements for unsupervised content discovery in audio. The minimum Bregman information principle is proposed in [4] as a generalization of the maximum entropy principle. Based on this principle, an algorithm for the Bregman co-clustering problem is developed. [5] adapted the Bregman co-clustering algorithm to a collaborative filtering framework. The key idea in this work is to simultaneously obtain user and item neighborhoods via co-clustering and generate predictions based on the average ratings of the co-clusters while taking into account the individual biases of the users and items. A well studied problem of co-clustering in data mining literature has been that of documents and words. In [6], a joint distribution is defined over words and documents to first find word-clusters that capture most of the mutual information about the set of documents, and then find document clusters, that preserve the information about the word clusters. Mandhani et al. [7] have proposed a two-step partitional-agglomerative algorithm to hierarchically co-cluster documents and words. Amongst non-hierarchical co-clustering, the methods proposed can mainly be grouped into the ones on matrix factorization, and graph partitioning. Long et al. [8] first proposed an approach to discover sub-matrices of a data matrix, based on matrix factorization. Using an iterative approach, sub-matrices of the original matrix were derived. Another popular approach that has been taken for co-clustering is treating it as a problem of partitioning bipartite graphs [9]–[11]. However, in spite of the above efforts, co-clustering of evolving data, has remained unaddressed.

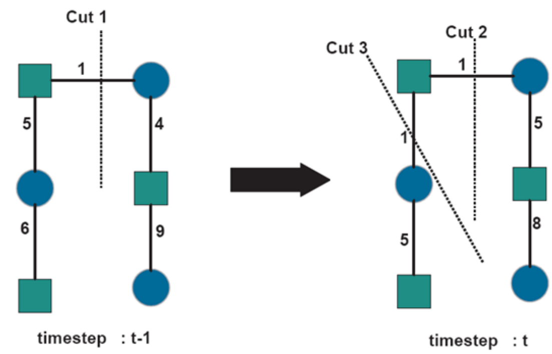

Figure 1. ESCC uses information from the previous time-step $(t-1)$ to maintain cluster membership into the present time (t) in order to provide a smoother transition. Here, we can seen that in order to maintain that membership, despite the evenly weighted cut in the present time, the past comes into effect to make **cut2** the better choice at time-step t.

In this paper, we present ESCC, a new approach for

evolutionary **s**pectral **c**o-**c**lustering. Consider the illustration shown in Figure 1. The two data types (i.e. instances and features) have been shown using the two shapes. Edge weights denote the level of association between them. At timestep $t-1$, assuming it is the first timestep, $cut1$ is appropriately chosen. At timestep t, the data has evolved, i.e. the association between instances and features has changed. Of the two cuts possible, ESCC will opt for $cut2$, as the clustering result does not move away from the recent past. The clustering decision is swayed by a historical cost function which incorporates information about previous timesteps. At every timestep, the resulting matrices formed from these cost functions, are first decomposed using spectral value decomposition (SVD) and the k-means algorithm [1] is then applied to obtain the desired number of clusters. In order to evaluate our approach, extensive experiments on synthetic and real world data have been performed.

II. Background and Related Work

In this section, we review concepts and literature relevant to the proposed approach.

A. Evolutionary Clustering

Clustering of evolving data (a.k.a. *Evolutionary clustering*) has been a relatively new topic and was first formulated by Chakrabarti et al. in [12]. They proposed heuristic solutions to evolutionary hierarchical clustering problems and evolutionary k-means clustering problems. Chi et al. [13] extended this work by proposing two evolutionary spectral clustering algorithms by incorporating a measure of *temporal smoothness* in the overall clustering quality. Evolutionary clustering differs from incremental clustering [14], which primarily addresses the issue of updating cluster centers [15], medoids [16] or hierarchical trees [17] when new data points arrive. Typically, the "new" arriving data points have no direct relationship with the "old" data points. Li et al. [18] have proposed an algorithm for clustering moving objects. The spacial-temporal regularities of the moving objects are discovered by using micro-clustering [19]. An incremental spectral clustering algorithm is proposed in [20] to cluster evolving data points. Both these works try to achieve higher computational efficiency by compromising on the clustering quality.

B. Spectral Co-Clustering

In this section, we provide a brief background on spectral co-clustering [9], [10]. Given an instance by feature data matrix W, the bipartite degree matrix is defined as,

$$D = \begin{bmatrix} D_1 & 0 \\ 0 & D_2 \end{bmatrix} \tag{1}$$

where $D_1(i,i) = \sum_j W_{ij}$ and $D_2(j,j) = \sum_i W_{ij}$.

Symbol	Definition
W_{mxn}	A data matrix of size m x n
W_t	W matrix for time-step t
W^T	Transpose of the given matrix
$W_{(i,:)}$, $W_{(:,i)}$	The row or column at i from matrix W, respectively
$W'_{wrt(W)}$	The matrix W' with respect to W
D	A diagonal degree matrix described in Section II-B
D_1, D_2	Diagonal degree matrices for instances and features, respectively
$D_{n,t}$	Degree matrix for the given dimension n and time-step t
k	Number of clusters
C_n	Cluster number n
t	Time-step
μ	Mean value
$\vec{d}$	A vector of indices having differences resulting from the comparison of two matrices
d	An index in $\vec{d}$
svd	The singular value decomposition function
α, β	Constants for tuning the algorithm
RTC|RTH	Choose RTC algorithm or RTH algorithm
Xu, Xv	Right and left singular vectors respectively ordered by descending singular value

Table I
Symbol Definitions

The bipartite Laplacian matrix is defined as,

$$L = \begin{bmatrix} D_1 & -W \\ -W^T & D_2 \end{bmatrix} \tag{2}$$

The partitions can then be obtained by solving the generalized eigenvalue problem $Lz = \lambda Dz$. However, due to the bipartite nature of the problem, the eigenvalue problem reduces to a much efficient SVD problem, as follows,

$$W_n = D_1^{-\frac{1}{2}} W D_2^{-\frac{1}{2}} \tag{3}$$

SVD is then performed on matrix W_n to get the left and right singular vectors. Applying k-means on these vectors, yields the co-clustering.

III. ESCC

We now present our evolutionary spectral co-clustering approach. Our algorithm is efficiently able to handle the case of insertions and deletions of instances, and features between timesteps, while demonstrating a resistance to changes between time-steps in the instances and features. Moreover, the algorithm handles cluster membership changes for instances and features between time-steps.

A. RTC vs. RTH

We propose two methods for including past time-step clustering into the present. In the rest of the paper, these methods are each referred to as the *evolutionary method* by which clusters are chosen from time-step to time-step. The first of these methods is Respect To the Current (RTC), wherein the present clustering quality (CQ) is of most importance and historical cost (HC) is calculated with only one previous time-step. The second, is Respect To Historical (RTH), which attempts to keep instances and features tied to the same clusters between time-steps, therefore this method uses all previous time-steps when calculating historical cost. For example, in Figure 1 where either evolutionary method would choose $cut2$, however if the weight of the edge on $cut2$ were higher, then the two would disagree. If making cuts with RTC, the optimal clustering for the present time will be chosen (i.e. $cut3$) within a margin of tolerance dictated by the values chosen for the constants, α and β, shown in Equation 4. On the other hand, when making cuts with RTH within the tolerance, $cut2$ would be chosen as it reduces the error in historical clustering.

$$EC_t = k - trace\left[Xv_t^T\left(\alpha CQ_t + \beta HC_t\right)Xu_t\right] \quad (4)$$

The accuracy of the clustering for each of these is determined by the cost of each time-step, known as the *evolutionary cost*. The evolutionary cost (EC) is computed through a summation of the clustering quality and the historical cost. Historical cost being the added negative weight for choosing a cut that causes a cluster change. RTC and RTH have differing cost functions as each are modeled for different purposes. The CQ and HC for RTC is defined as,

$$CQ_t = \alpha D_{1,t}^{-\frac{1}{2}} W_t D_{2,t}^{-\frac{1}{2}} \quad (5)$$

$$HC_t = \beta D_{1,t-1}^{-\frac{1}{2}} W_{t-1} D_{2,t-1}^{-\frac{1}{2}} \quad (6)$$

CQ_t refers to the cluster quality at time t, while HC_t refers to the historical cost at time t. These measures differ in that CQ measures the quality of the present clustering as if it were static data and CH measures the cost of change from the previous time-step to the present. Similarly, the CQ and HC for RTH is defined as,

$$CQ_t = \alpha D_{1,t}^{-\frac{1}{2}} W_t D_{2,t}^{-\frac{1}{2}} \quad (7)$$

$$HC_t = \beta Xu_{t-1} Xv_{t-1}^T \quad (8)$$

As Xu and Xv are created from the left and right singular vectors from all previous time-steps, the HC for RTH is more tightly connected to the past clustering choices. In contrast, the HC for RTC is only concerned with values from the last time-step alone.

The aim is to minimize these costs to give the best possible cut through the data. The minimization is achieved by using the top k singular vectors from the left (Xu_t) and the right (Xv_t), after performing SVD, as explained next.

B. Piecing the algorithm together

Depending on the evolutionary method chosen, the computation of Xu and Xv varies. For RTC, we obtain the singular vectors as,

$$[Xu_t, Xv_t] = svd\left[\alpha W_t + \beta D_{1,t-1}^{-\frac{1}{2}} W D_{2,t-1}^{-\frac{1}{2}}\right] \quad (9)$$

For RTH, Xu and Xv, are obtained as,

$$[Xu_t, Xv_t] = svd\left[\alpha W_t + \beta Xu_{t-1} Xv_{t-1}^T\right] \quad (10)$$

wherein all references to time-step t are of the form t with respect to the current time-step. For example, if there is an instance increase between time-step $t-1$ and t, $t-1$ with respect to t would have the additional instances as a balanced insertion to the matrix. The collected singular vectors are passed to k-means in order to generate clusters for each time-step. The final matrix Z consisting of all the singular vectors is given by,

$$Z = \begin{bmatrix} D_{1,t}^{-\frac{1}{2}} & Xu_t(2..\lceil \log_2 k \rceil) \\ D_{2,t}^{-\frac{1}{2}} & Xv_t(2..\lceil \log_2 k \rceil) \end{bmatrix} \quad (11)$$

where Xu_t and Xv_t are comprised of all left and right singular vectors, respectively, for time-step t, and k is the number of clusters. The result of putting all of this together is presented in the ESCC algorithm, shown in Algorithm 2.

C. Handling Data Changes

The balanced insertion discussed previously is meant to make the dimensions of each time-step comparable while not compromising the cluster assignments of other instances. This need to handle changes in the data dimensions over time requires a new construct, called the WRT construct.

Definition: Given two matrices, H and C, existing on the same time-line in different time-steps, then H is a WRT Matrix to $C \iff$ the instances and features found in H are equivalent to those found in C. If this is not the case H can be made a WRT Matrix by use of Algorithm 1.

In order to accomplish this, when adding instances or features to the past, different methods are used based on the selection of RTC or RTH. In the case of RTC, where present quality is most important, the inserted row or column receives the average of the whole matrix at present time for each cell. For RTH, the inserted row or column receives the average of the new row or column in each cell.

The easier of the two operations is removal of instances or features from past time-steps. This occurs when an instance or column in the past matrices is no longer present in the current time-step. In this case, the instance or column can be removed from the past in each WRT Matrix.

IV. Experiments

The proposed approach has been evaluated on synthetic as well as real world data. The synthetic dataset is designed to show the different functions of the algorithm, while the accuracy of the algorithm is evaluated on real world data from the popular PubMed database.

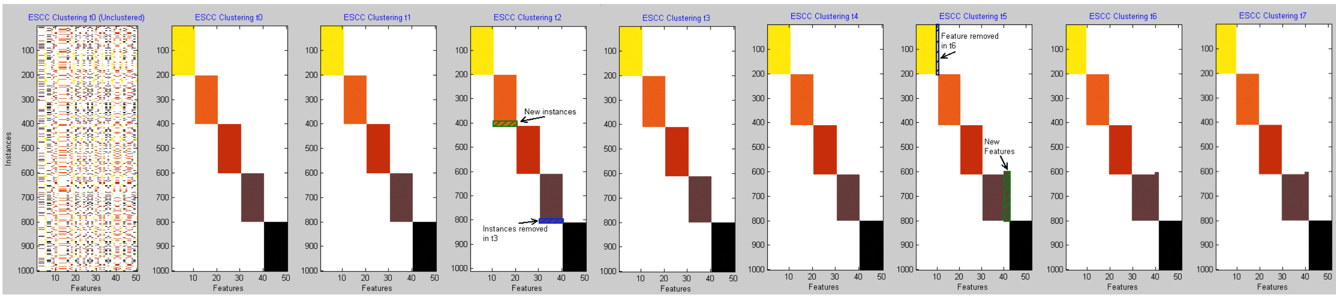

Figure 2. t_0 is the original data in unclustered and clustered form. t_1 shows consistency of clustering despite 50 instances of noise per cluster. t_2 shows 10 instances added to C_2. t_3 shows 10 instances removed from C_4 (highlighted in t_2). t_4 shows cluster stability through a time-step of no change. t_5 shows 2 features added to C_4. t_6 shows 1 feature removed from C_1 (highlighted in t_5). t_7 shows again the stability through an unchanged time-step.

A. Synthetic Data

Three different synthetic datasets were used to show the features of the algorithm. The first, shown in Section IV-A1, demonstrates the algorithm's ability to handle noise, and additional and removal of instances and features. The second (Section IV-A2) synthetic data set shows the algorithm's ability to track instances through cluster shifts over time. Finally, the third (Section IV-A3) synthetic data set shows the algorithm's ability to track features through cluster shifts over time.

1) Synthetic Demonstrations: We constructed a synthetic dataset with 8 time-steps and 5 clusters of 200 instances, each having 10 assigned features. The initial time-step and 5 clusters were formed by creating 5 ascending groups of data sampled from 5 normal distributions. Each normal distribution was guaranteed to be distanced from the previous through an augmented μ value added to the previous μ. Therefore, if a cluster, C_n, has an assigned μ it is represented by μ_n where n is the cluster number, then the distributions are determined by $\mu_n = \mu_{n-1} + 5 \cdot n + R$ where R is a random integer bound by $[0 - 100]$. For all distributions $\sigma = 1$.

To form time-step t_1 Gaussian noise was added to 50 instances per cluster. As shown in Figure 2, t_0 and t_1 are unchanged, despite the added noise. Next, in t_2, instances were added to C_2 using a selection of values from a normal distribution having a $\mu = \mu_2$. The figure shows the new instances have been appropriately clustered as indicated by the green shaded box. To show the opposing action, in t_3, instances are removed from C_4. From the figure, it can be seen that all clusters remain unchanged and C_4 has a smaller block size. This was also indicated in t_3 as the instances that will be removed are highlighted in blue. In t_4, the data is unchanged, simulating the stability of the clustering between time-steps.

Until this point the algorithm has behaved as any previous evolutionary algorithm would. Next, with t_5, features are added to the dataset in C_4 using values selected from a normal distribution having a $\mu = \mu_4$. The figure shows that the additional features have not affected the clusters and the resulting new columns were correctly clustered as indicated by the green box around the column of data in Figure 2, time-step t_5. In t_6, features were removed from C_1 and the figure shows again that the clustering is unaffected and the block for C_1 is diminished. This was also indicated in t_5 by the blue shading around the feature column of C_1. Finally, the data is left untouched for t_7 as evident in the figure.

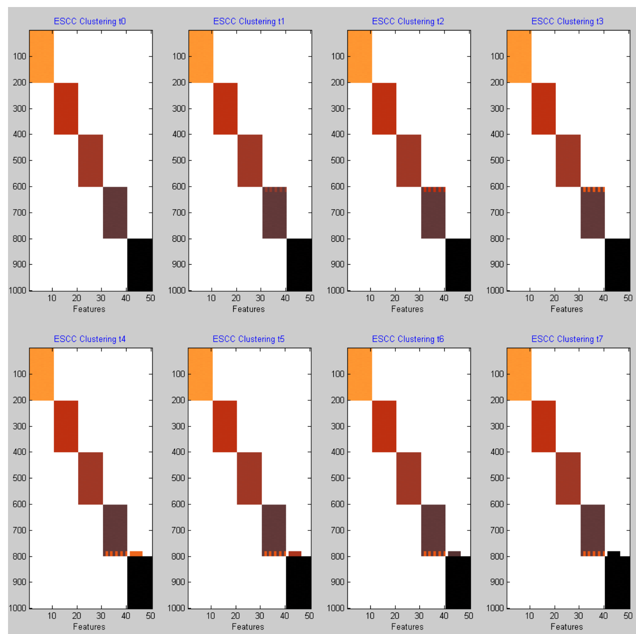

Figure 3. t_0 is the original data from Figure 2. t_1 - t_3 show the downward shift of the C_4 features and t_4 - t_7 show the up shift in C_5 features. Ultimately the 20 instances were shown to shift from C_4 to C_5 as expected.

2) Synthetic Instance Drift: Using the same initial time-step and 5 cluster distribution from section IV-A1, a group of 20 instances were incrementally modified at each time-step, up to 8, to denote a cluster shift of the instances. Figure 3 shows the results of the clustered data. As we can see, at t_3 the instances are discolored denoting a drop in value, and in t_4 the instances change clusters when values for the next cluster's features begin rising.

Algorithm 1 Generate WRT Matrices Algorithm

INPUT: $W'_{1:t}$(history), W_t(present), RTC|RTH { // $history$ contains all weight matrices previous to $present$, $present$ contains the current time-step for which this WRT matrix is being created, RTC|RTH is the evolutionary method chosen }
OUTPUT: $W_{wrt:t}$
METHOD: { // Start with removal of differing rows and columns }
for $t' \leftarrow 1..$ length of history **do**
  $\vec{d} \leftarrow W'_{t'} \notin W$
  $W_{t'_{wrt(W_t)}} \leftarrow W'_{t'} - W_{t,(\vec{d},:)}$
  $\vec{d} \leftarrow W'^{T}_{t} \notin W^T$
  $W_{t'_{wrt(W_t)}} \leftarrow \left(W'^{T}_{t'} - W^{T}_{t,(:,\vec{d})}\right)^T$
end for
{ // Make additions next, these will be based on the evolutionary method } { // Start with Row/Instance addition }
for $t' \leftarrow 1..$ length of history **do**
  $\vec{d} \leftarrow W \notin W'_{t'}$
  if RTC **then**
    for all differences $\vec{d}$ **do**
      $insert \leftarrow \mu\left(W'_{t'}\right)$
      Add $insert$ to $W_{t'_{wrt(W_t)},(d,:)}$ based on its ID
    end for
  else if RTH **then**
    for all differences $\vec{d}$ **do**
      $insert \leftarrow \mu\left(W_{t',(\vec{r_d})}\right)$
      Add $insert$ to $W_{t'_{wrt(W_t)},(d,:)}$ based on its ID
    end for
  end if
end for
{ // Column/Feature addition next }
for $t' \leftarrow 1..$ length of history **do**
  $\vec{d} \leftarrow W \notin W'_{t'}$
  if RTC **then**
    for all differences $\vec{d}$ **do**
      $insert \leftarrow \mu\left(W'_{t'}\right)$
      Add $insert$ to $W_{t'_{wrt(W_t)}}$ based on its ID
    end for
  else if RTH **then**
    for all differences $\vec{d}$ **do**
      $insert \leftarrow \mu\left(W_{t',(\vec{c_d})}\right)$
      Add $insert$ to $W_{t'_{wrt(W_t)},(:,d)}$ based on its ID
    end for
  end if
end for

Algorithm 2 ESCC Algorithm

INPUT: W_{mxnxt}, k, and RTC|RTH { // W is an m x n x t matrix where t is the number of time-steps, k is the number of clusters }
OUTPUT: Cluster assignments for each time-step
METHOD:
for all time-steps **do**
  for all time-steps previous to the current **do**
    create corresponding past matrices with respect to the present time-step using Algorithm 1
  end for
  for all time-steps previous to the current **do**
    if RTC **then**
      use equation 9 for SVD
    else if RTH **then**
      use equation 10 for SVD
    end if
    combine left and right singular vector matrices using equation 11
    run k-means on resulting matrix
  end for
end for{ // The first m cluster values for each time-step represents the instance clustering and the trailing n cluster values for each time-step represent the feature clustering.}

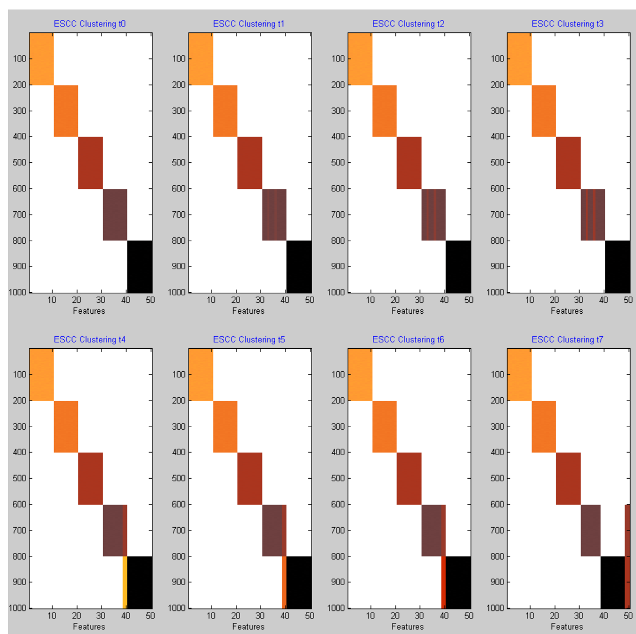

Figure 4. t_0 is the original data from Figure 2. t_1 - t_3 show the downward shift of the C_4 instances at 2 distinct features and t_4 - t_7 show the up shift in C_5 instances at those features. Ultimately the 2 features were shown to shift from C_4 to C_5 as expected.

3) Synthetic Feature Drift: In the same manner as the previous example, a feature shift is shown in Figure 4. Two

Year	# Papers	# Authors	# Words
2000	2817	4907	6750
2001	2792	5254	6860
2002	3486	6220	7117
2003	3698	6734	7205
2004	4059	7244	7327
2005	4059	8023	7452
2006	4376	7984	7286
2007	4690	8365	7302
2008	4962	8234	7328
2009	5074	7988	7304

Table II
AN OUTLINE OF THE PUBMED DATASET.

features were chosen from C_4 to have their values lowered in C_4 and raised in C_5. As with the last experiment, the color change indicates diminished values and at t_3 the values correlated with C_5 begin increasing and the cluster shift occurs. In t_7, it can be seen that the full shift has occurred.

B. Real Data

To evaluate the accuracy of ESCC on real data, we selected a widely used database of medical papers: PubMed[1]. The PubMed dataset was constructed from two searches of highly researched topics in the medical field: schizophrenia treatment and stem cell research. Each search was limited to English texts published between 1990 and 2009 having authors and abstracts. The papers were then parsed by year to obtain author-word matrices for each year containing both subjects. Common stop words were removed and all words were stemmed. Authors and words were assigned unique IDs tied to the first occurrence based on the year the author or word was used and the subject matter, respectively. These unique ids are used in the generation of the WRT matrices as demonstrated in Algorithm 1. Authors that had published less than three papers in the time-span were removed from the dataset and words occurring less than 30 times throughout all the abstracts were also removed. This resulted in 10 data time-steps consisting of information from 64,320 papers, 17,731 unique authors, and 8,541 unique words, as described in Table II.

Each year represented was run using the RTC and RTH approaches. The α and β values used were 0.7 and 0.3, respectively. The following series of confusion matrices and top words from the respective clusters show the ability for ESCC to co-cluster authors and words through each time-step. As can be seen in Figure 5, RTH outperforms RTC in nearly each time-step. This is mainly due to the fact that authors do not often change subjects, despite the enormous amount of noise in the words utilized in each abstract. Furthermore, Figure 6 shows that RTH maintains historical cluster membership better than RTC.

The results show that the algorithm is able to split the authors successfully, given the related studies within the

[1] http://www.ncbi.nlm.nih.gov/pubmed

RTC	SchizR	StemCell
Auth1:	1558	432
Auth2:	52	2864
RTH		
Auth1:	1558	432
Auth2:	52	2864

W1:	disord drug mental psychiatri stress
W2:	allogen repopul graftversushost leukem posttranspl

Table III
ESCC CLUSTERING OF PUBMED. YEAR 2000.

RTC	SchizR	StemCell
Auth1:	1805	374
Auth2:	609	2465
RTH		
Auth1:	1706	473
Auth2:	52	3022

RTC	
W1:	treatment schizophrenia disord depart human
W2:	colonyform gvh osteoclast transfus leukem
RTH	
W1:	treatment effect schizophrenia disord psychiatr
W2:	marrow bone transplant blood hematopoiet

Table IV
ESCC CLUSTERING OF PUBMED. YEAR 2001.

RTC	SchizR	StemCell
Auth1:	1848	578
Auth2:	79	3714
RTH		
Auth1:	1834	592
Auth2:	75	3718

RTC	
W1:	treatment effect schizophrenia univers depart
W2:	patient studi transplant bone stem
RTH	
W1:	treatment schizophrenia symptom clinic antipsychot
W2:	patient transplant bone stem blood

Table V
ESCC CLUSTERING OF PUBMED. YEAR 2002.

RTC	SchizR	StemCell
Auth1:	2199	613
Auth2:	149	3772
RTH		
Auth1:	2172	640
Auth2:	67	3854

RTC	
W1:	treatment schizophrenia result disord ptsd
W2:	patient studi bone univers depart
RTH	
W1:	treatment schizophrenia result disord ptsd
W2:	patient studi bone transplant univers

Table VI
ESCC CLUSTERING OF PUBMED. YEAR 2003.

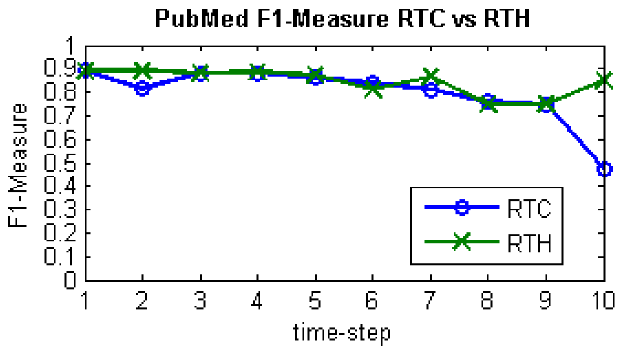

Figure 5. F-measures for the ESCC clustering on PubMed data. The higher the F-measure the higher the accuracy of the clustering. This shows that RTH is generally better at sorting out the different clusters.

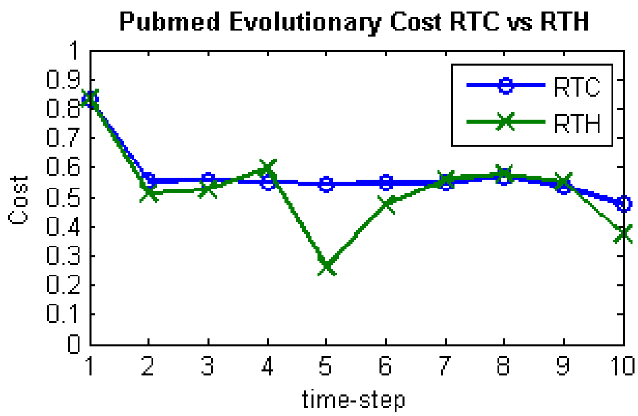

Figure 6. Using the cost formulas from Section III-A the two algorithms are measured for accuracy and maintaining historical clustering. A lower cost value indicates a better clustering in regard to maintaining history. As can be seen in the graph, we find that RTH generally performs better than RTC.

medical field. The RTH algorithm shows a better clustering through the end while the RTC algorithm did not finish as well as shown in Table XII. As more authors are introduced to the set, the words are fairly constant because of the constraint that each be used 35 or more times within the selected abstracts. This meant that the clusters became more saturated and blended the lines making the distinctions more difficult over time. However, by the evolutionary cost graph in Figure 6 where a maximum cost has a value of 2, the given costs show low changes in cluster membership, reducing noise. Observing the selection of top words clustered with each set shows that the co-clustering element of this algorithm is properly separating the words to associate with the correct authors.

V. Conclusions

We presented a new framework for co-clustering evolving data, under a spectral clustering paradigm. Under this framework, we discussed two approaches for incorporating historical information into the clustering results. Experiments on synthetic as well as real world data demonstrate the effectiveness of the proposed approach.

RTC	SchizR	StemCell
Auth1:	2492	657
Auth2:	306	3788
RTH		
Auth1:	2349	755
Auth2:	118	3976

RTC	
W1:	patient treatment result effect disord
W2:	studi bone transplant univers clinic
RTH	
W1:	patient treatment disord symptom antipsychot
W2:	studi bone transplant univers clinic

Table VII
ESCC clustering of PubMed. Year 2004.

RTC	SchizR	StemCell
Auth1:	2740	674
Auth2:	620	3988
RTH		
Auth1:	2665	749
Auth2:	746	3862

RTC	
W1:	treatment disord group symptom antipsychot
W2:	cell patient studi bone result
RTH	
W1:	patient studi treatment schizophrenia mai
W2:	cell bone result marrow transplant

Table VIII
ESCC clustering of PubMed. Year 2005.

References

[1] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM Comput. Surv.*, vol. 31, no. 3, pp. 264–323, 1999.

[2] I. S. Dhillon, S. Mallela, and D. S. Modha, "Information-theoretic co-clustering," in *proc. of ACM KDD*, 2003.

RTC	SchizR	StemCell
Auth1:	2784	718
Auth2:	792	3689
RTH		
Auth1:	2635	856
Auth2:	160	4332

RTC	
W1:	studi schizophrenia result ptsd effect
W2:	patient treatment bone stem marrow
RTH	
W1:	patient treatment univers symptom clinic
W2:	cell studi bone result stem

Table IX
ESCC clustering of PubMed. Year 2006.

RTC	SchizR	StemCell
Auth1:	3074	641
Auth2:	1395	3254
RTH		
Auth1:	3008	719
Auth2:	1400	3237

RTC	
W1:	treatment effect disord depart clinic
W2:	allogen epc gvhd osteoblast hmsc
RTH	
W1:	treatment effect disord clinic symptom
W2:	studi bone result univers group

Table X
ESCC CLUSTERING OF PUBMED. YEAR 2007.

RTC	SchizR	StemCell
Auth1:	2942	810
Auth2:	1273	3208
RTH		
Auth1:	2996	756
Auth2:	1325	3156

RTC	
W1:	treatment result disord ptsd univers
W2:	epc bmsc engraft gvhd graftversushost
RTH	
W1:	patient treatment result disord ptsd
W2:	cell studi bone effect univers

Table XI
ESCC CLUSTERING OF PUBMED. YEAR 2008.

RTC	SchizR	StemCell
Auth1:	3589	2
Auth2:	4381	15
RTH		
Auth1:	2587	1004
Auth2:	133	4263

RTC	
W1:	patient studi treatment schizophrenia ptsd
W2:	colonyform thalidomid feeder bdlsc gfplabel
RTH	
W1:	patient treatment disord group clinic
W2:	cell studi result effect msc

Table XII
ESCC CLUSTERING OF PUBMED. YEAR 2009.

[3] R. Cai, L. Lu, and A. Hanjalic, "Unsupervised content discovery in composite audio," in *proc. of ACM MM*, 2005.

[4] A. Banerjee, I. Dhillon, J. Ghosh, S. Merugu, and D. S. Modha, "A generalized maximum entropy approach to bregman co-clustering and matrix approximation," *Journal of Machine Learning Research*, vol. 8, pp. 1919–1986, 2007.

[5] T. George and S. Merugu, "A scalable collaborative filtering framework based on co-clustering," in *proc. of IEEE ICDM*, 2005.

[6] N. Slonim and N. Tishby, "Document clustering using word clusters via the information bottleneck method," in *proc. of ACM SIGIR*, 2000, pp. 208–215.

[7] B. Mandhani, S. Joshi, and K. Kummamuru, "A matrix density based algorithm to hierarchically co-cluster documents and words," in *proc. of ACM WWW*, 2003, pp. 511–518.

[8] B. Long, Z. M. Zhang, and P. S. Yu, "Co-clustering by block value decomposition," in *proc. of ACM KDD*, 2005.

[9] I. S. Dhillon, "Co-clustering documents and words using bipartite spectral graph partitioning," in *proc. of ACM KDD*, 2001.

[10] H. Zha, X. He, C. H. Q. Ding, M. Gu, and H. D. Simon, "Bipartite graph partitioning and data clustering," in *proc. of ACM CIKM*, 2001.

[11] M. Rege, M. Dong, and F. Fotouhi, "Co-clustering documents and words using bipartite isoperimetric graph partitioning," in *proc. of IEEE ICDM*, 2006.

[12] D. Chakrabarti, R. Kumar, and A. Tomkins, "Evolutionary clustering," in *proc. of ACM KDD*, 2006.

[13] Y. Chi, X. Song, D. Zhou, K. Hino, and B. L. Tseng, "Evolutionary spectral clustering by incorporating temporal smoothness," in *proc. of ACM KDD*, 2007.

[14] W. Pedrycz and K.-C. Kwak, "The development of incremental models," *IEEE Transactions on Fuzzy Systems*, vol. 15, no. 3, pp. 507–518, June 2007.

[15] C. Gupta and R. Grossman, "Genic: A single pass generalized incremental algorithm for clustering," in *proc. of SDM*, 2004.

[16] S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan, "Clustering data streams," in *proc. of IEEE Symposium on Foundations of Computer Science*, 2000.

[17] M. Charikar, C. Chekuri, T. Feder, and R. Motwani, "Incremental clustering and dynamic information retrieval," in *in proc. of the 29th STOC Conference*, 1997.

[18] Y. Li, J. Han, and J. Yang, "Clustering moving objects," in *proc. of the 10th ACM SIGKDD Conference*, 2004.

[19] T. Zhang, R. Ramakrishnan, and M. Livny, "Birch: An efficient data clustering method for very large databases," in *proc. of ACM ACM SIGMOD*, 1996, pp. 103–114.

[20] H. Ning, W. Xu, Y. Chi, Y. Gong, and T. Huang, "Incremental spectral clustering with application to monitoring of evolving blog communities," in *proc. of SDM*, 2007.