

Co-Clustering by Similarity Refinement

Jian Zhang
SRI International
333 Ravenswood Ave
Menlo Park, CA 94025

Abstract

When a data set contains objects of multiple types, to cluster the objects of one type, it is often necessary to consider the cluster structure on the objects of the other types. Co-clustering the related objects often generates better clusters. One basic connection here is that the similarity among the objects of one type is often affected by the cluster structures on the objects of the other types. Although many co-clustering schemes have been proposed, none has explicitly explored such a connection.

We propose a framework that utilizes this connection directly. In this framework, employing a spectral-embedding-based approach, we first obtain certain approximate cluster information about the objects of individual types. Such information is then used to refine the similarity measures for the objects of the related types. The final clustering is performed with the refined similarity. We tested our framework on both bipartite-graph and document clustering. Our experiments showed that the refined similarity leads to much better clustering, indicating that our refinement makes the similarity measures closer to their true values.

1 Introduction

In recent years, co-clustering has been intensively studied in the machine learning community [1, 8, 5]. In co-clustering, the data to be clustered involve objects of different types that are related to each other, for example, documents and terms in a corpus, customers and items in collaborative filtering.

Clustering aims to cluster objects in the data set according to the similarity between the objects. Given a collection of different types of objects, the problem considered by co-clustering requires clustering the objects of the same type. One may take a simple approach to this problem, ignoring the relationships between different types of objects and clustering each type of object separately. However, because the relationship information is ignored, this approach

may produce poor clusters. Particularly, when the cluster structure on one type of object can affect the similarity between the objects of the other types, considering objects of different types independently loses information that may be essential for clustering.

For example, in the term-document model, the similarity of the documents is determined by the term vectors that represent them. If one document uses the term “paper” to refer to published work while the other uses “literature”, although the semantics of the two terms are close in this context, they will not contribute to the similarity measure of the two documents. On the other hand, if we know the cluster structure that puts the two terms in one cluster, we can relate the two terms when calculating the similarity measure for the two documents.

This shows the necessity to cluster the objects of different types simultaneously as the clustering of one type of object is connected to that of the other types. Although many co-clustering schemes have been proposed [1, 8, 5, 4, 2], none of these has explicitly explored the connection between the similarity measure for one type of object and the cluster structures on the objects of the other types.

Most earlier co-clustering algorithms fall into three categories. One category models each type of object as a random variable, clustering the objects of different types simultaneously while preserving the mutual information between the random variables that model these objects [1]. The second category models the relationship between different types of objects as a (nonnegative) matrix. This matrix is *approximately* decomposed into several matrices, which indicate the cluster memberships for the objects [4, 5, 2]. The third category treats the relationship between different types of objects as a graph and performs co-clustering by graph partitioning based on spectral analysis [9].

We observe that these schemes do not explicitly tie together the similarity measure for one type of object and the cluster structures on the objects of the other types. The difficulty here is that before clustering, we do not know the true cluster structure on objects of any type. Identifying these clusters is exactly the goal of our clustering process.

It seems that we are in a chicken-egg situation. However, we can get around this difficulty by using some approximate cluster information rather than the exact true cluster structures. The approximate cluster information on one type of object can then be utilized to refine the similarity measures between the objects of the other types. We call this process *similarity refinement*.

In this paper, we introduce a similarity refinement framework. In particular, we use spectral embedding to obtain a refinement matrix. The matrix approximates the cluster structure in the sense that if two objects belong to the same clusters, the corresponding entry in the matrix would be close to one, and otherwise the entry would be close to zero. There may be noises in this matrix, and therefore it is only an approximation of the cluster structure. The framework then uses this matrix to refine the similarity between the objects of the other types. The final clustering is derived from the refined similarity measures. In this way, we explicitly explore the (approximate) cluster structure on one type of object when calculating the similarity for the objects of the other types.

We tested our co-clustering on both bipartite-graph and document clustering. Our experiments showed that similarity refinement based on approximate cluster information can provide a similarity estimation closer to the true measure, and hence leads to better clustering.

The rest of the paper is organized as follows: In Section 2, we present the details of our similarity refinement framework. We show how the approximate cluster information can be obtained. We then focus on similarity refinement using this information for both vector and set representations of objects. We discuss our experiments in Section 3. Section 4 presents our conclusions.

2 A Similarity Refinement Framework

The goal of the framework is to refine the similarity between objects of one type using approximate cluster information on the objects of the related types. We will focus on problems that involve two types of objects, but the refinement framework can be extended to problems involving multiple types of objects.

We denote by $A = \{a_1, a_2, \dots, a_n\}$ and $B = \{b_1, b_2, \dots, b_m\}$ the two types of objects, respectively (e.g., documents and terms). For two objects of the same type, their similarity measure involves objects of the other type. In many cases, these objects are often represented using the objects of the other type. For example, a document can be represented by the set of terms it contains, and a customer can be represented by the set of items he or she purchases. Let sim^A be the function that measures similarity for the objects in A and sim^B the similarity function for the objects in B . Traditionally, these similarity functions have the

following format: $sim^X(\cdot, \cdot) : X \times X \rightarrow \mathbb{R}$ where X can be A or B . The framework we introduce will extend this function so that when computing the similarity between objects in A , we can also consider the clustering structure on objects in B , and vice versa.

Formally, we would like to define two refinement matrices \mathbf{R}_A and \mathbf{R}_B such that sim^A becomes $A \times A \times \{\mathbf{R}_B\} \rightarrow \mathbb{R}$ and sim^B becomes $B \times B \times \{\mathbf{R}_A\} \rightarrow \mathbb{R}$. We refer to this extended similarity function as “ \overline{sim} ”. In the ideal situation, if we know the true cluster structure on the objects in B , we can construct \mathbf{R}_B as follows: $R_B(i, j) = 1$ if b_i and b_j belong to the same cluster and $R_B(i, j) = 0$ otherwise. For two objects $a_u, a_v \in A$, $\overline{sim}^A(a_u, a_v, \mathbf{R}_B)$ can then take advantage of the cluster structure among the objects b_i . Unfortunately, the true cluster structure is unknown. To overcome this difficulty, we relax the definition of the refinement matrices. We would like $R_B(i, j)$ to be close to one if b_i and b_j are most likely in the same cluster and close to zero if not. We use spectral embedding to obtain such a refinement matrix that approximates the ideal matrix.

Given the set of objects A and their pairwise similarity matrix \mathbf{S}_A (assuming that the similarity function is non-negative and symmetric, i.e., $sim(x, y) = sim(y, x) \geq 0$), spectral embedding first computes the k eigenvectors with the largest eigenvalues of the matrix \mathbf{S}_A . Let $\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_k$ be the eigenvectors. The objects are then embedded into a k -dimensional space where object a_i has the coordinate that is the normalized version of the vector $\langle e_1(i), e_2(i), \dots, e_k(i) \rangle$ (we refer to the i -th element of the vector \mathbf{e} by $e(i)$). We denote by $\mathcal{E}(a_i)$ the coordinate vector of the object a_i in the embedding space. Spectral embedding maps each object to a point on the unit sphere. It has been shown in [7] that the inner product between the mapped points tends to be one if the corresponding objects fall in the same cluster. Otherwise, the inner product tends to be zero. When the similarity measure is not perfect with respect to the true clustering, the inner product would not be perfect ones and zeros, but the general trends still hold. Considering this, we set a threshold $0 < \alpha < 1$. Only when the inner product value is above the threshold, we will use it in the refinement matrix. Otherwise, we simply set the corresponding entry to be zero.

Note that to compute the spectral embedding of the objects, all we need is the similarity matrix \mathbf{S} , which can be obtained using sim . (\overline{sim} is not required here.) The general co-clustering scheme in our framework then goes as follows: We compute a pairwise similarity matrix for objects in A and B , respectively, and then obtain the spectral embedding of these objects. The refinement matrices \mathbf{R}_A and \mathbf{R}_B are obtained by letting $R_A(i, j) = (\mathcal{E}(a_i) \otimes \mathcal{E}(a_j))_{\alpha+}$ and $R_B(i, j) = (\mathcal{E}(b_i) \otimes \mathcal{E}(b_j))_{\alpha+}$, respectively, where \otimes represents the inner product and $(x)_{\alpha+}$ is the thresholding function, i.e. $(x)_{\alpha+} = x$ for $x \geq \alpha$ and

$(x)_{\alpha+} = 0$ for $x < \alpha$. Finally, we update \mathbf{S}_A and \mathbf{S}_B by letting $S_A(i, j) = \overline{sim}^A(a_i, a_j, \mathbf{R}_B)$ and $S_B(i, j) = \overline{sim}^B(b_i, b_j, \mathbf{R}_A)$. Once we obtain the refined similarity matrices \mathbf{S}_A and \mathbf{S}_B , we compute the spectral embedding using these refined matrices and cluster the objects by standard clustering methods such as kmeans in the spectral embedding space. Fig. 1 summarizes this process.

1. Compute similarity matrices:
 $S_A(i, j) \leftarrow sim^A(a_i, a_j)$
 $S_B(i, j) \leftarrow sim^B(b_i, b_j)$
2. Obtain $\forall a_i \in A, \mathcal{E}(a_i)$ from S_A
and $\forall b_i \in B, \mathcal{E}(b_i)$ from S_B .
3. Construct refinement matrices:
 $R_A(i, j) \leftarrow (\mathcal{E}(a_i) \otimes \mathcal{E}(a_j))_{\alpha+}$
 $R_B(i, j) \leftarrow (\mathcal{E}(b_i) \otimes \mathcal{E}(b_j))_{\alpha+}$
4. Recompute similarity:
 $S_A(i, j) \leftarrow \overline{sim}^A(a_i, a_j, \mathbf{R}_B)$
 $S_B(i, j) \leftarrow \overline{sim}^B(b_i, b_j, \mathbf{R}_A)$
5. Clustering using \mathbf{S}_A and \mathbf{S}_B .

Figure 1. Similarity-Refinement-Based Co-Clustering Scheme

One may repeat steps 2 through 4 in Fig. 1 a few times to further refine the similarity. Our experiments show that in many cases, the change to the similarity measure made by the first refinement is the largest. The effect of the second refinement is often 1/5-1/10 compared to the first one. And the similarity change caused by the following refinements are even smaller. Therefore, one refinement accounts for most of the improvement. Further refinement can still make improvements but the improvements are small compared to the first one.

Given a similarity function \overline{sim} that takes advantage of the refinement matrix, we can apply the scheme in Fig. 1 and obtain a similarity-refinement-based co-clustering. In the rest of this section, we consider two types of similarity measures and define \overline{sim} for these measures.

2.1 Similarity Refinement for Vector Representation

Representing objects in the data set as normalized vectors is a common practice. The similarity between two objects is defined to be the inner product of the two corresponding vectors. For example, documents are often represented by normalized $tf \times idf$ vectors. In this subsection,

we will use this example to describe how to refine similarity. Note that this refinement can be applied to clustering objects other than document as long as the (normalized) vector representation is used.

Let \mathbf{a}_i be the vector representing the document a_i . Note that $a_i(j)$ corresponds to the j -th term. To simplify the description, in what follows, we will use “element” and “term” in an interchanging fashion. For example, instead of saying that the terms that these vector elements correspond to belong to the same cluster, we would simply say that the “elements” belong to the same cluster.

Consider two documents represented by normalized $tf \times idf$ vector \mathbf{a}_i and \mathbf{a}_j . Let s and t be two different terms. Suppose they are closely related and belong to the same cluster according to their appearance patterns in the collection of documents. To measure the similarity between documents i and j , we would like to have $a_i(s) \cdot a_j(t)$ (and $a_i(t) \cdot a_j(s)$) contribute to this measure. For this purpose, we use the refinement matrix to transform \mathbf{a}_i and \mathbf{a}_j such that the elements that belong to the same cluster are aligned after the transformation.

More formally, following the steps in Fig. 1, from the term similarity matrix, we obtain the embeddings of the terms, using which the refinement matrix \mathbf{R} is constructed. We then transform document \mathbf{a}_i to $\hat{\mathbf{R}} \cdot \mathbf{a}_i$, where $\hat{\mathbf{R}}$ is the normalized version of \mathbf{R} (i.e., we obtain $\hat{\mathbf{R}}$ by normalizing the 2-norm of each column of the matrix \mathbf{R}). For two documents \mathbf{p}, \mathbf{q} , the similarity measure after refinement is then $\hat{\mathbf{R}} \cdot \mathbf{p} \otimes \hat{\mathbf{R}} \cdot \mathbf{q}$, i.e., $\overline{sim}(\mathbf{p}, \mathbf{q}, \mathbf{R}) = \hat{\mathbf{R}} \cdot \mathbf{p} \otimes \hat{\mathbf{R}} \cdot \mathbf{q}$.

We now give a simple analysis on this refinement. We abuse the notation and let \mathbf{p}, \mathbf{q} be the corresponding $tf \times idf$ vector for the two documents. The representation vectors for the two documents are now $\frac{\mathbf{p}}{\|\mathbf{p}\|_2}$ and $\frac{\mathbf{q}}{\|\mathbf{q}\|_2}$. Suppose the vectors are k -dimensional and the terms in the documents all belong to the same cluster. (Note that $p(i)$ may not be equal to $q(i)$. One may imagine the following scenario: we have two documents that mean exactly the same thing. One uses a set of words and the other uses a different set, but the two sets of words are semantically equal.) Although we consider only the case where all terms belong to the same cluster in this example, similar analysis applies to cases where more than one clusters are involved.

Let $x = \sum p(i)$ and $y = \sum q(i)$. In the ideal situation, we would like to represent document p by the 1-dimensional vector $\langle \frac{x}{\sqrt{x^2}} \rangle = \langle 1 \rangle$ and document q by $\langle \frac{y}{\sqrt{y^2}} \rangle = \langle 1 \rangle$. The similarity measure is then $\frac{xy}{\sqrt{x^2 y^2}} = 1$. However, we do not know the cluster of the terms. Rather, we calculate the refined similarity as $\hat{\mathbf{R}} \cdot \frac{\mathbf{p}}{\|\mathbf{p}\|_2} \otimes \hat{\mathbf{R}} \cdot \frac{\mathbf{q}}{\|\mathbf{q}\|_2} = \frac{\mathbf{p}^T \hat{\mathbf{R}}^T \hat{\mathbf{R}} \mathbf{q}}{\|\mathbf{p}\|_2 \|\mathbf{q}\|_2}$, where T means the transpose of the matrix. Let $\hat{R}(i)$ be the i -th column of the matrix $\hat{\mathbf{R}}$. The above quantity can be rewritten as

$\frac{\mathbf{p}^T \hat{\mathbf{R}}^T \hat{\mathbf{R}} \mathbf{q}}{\|\mathbf{p}\|_2 \|\mathbf{q}\|_2} = \sum_i \sum_j (\hat{R}(i) \otimes \hat{R}(j)) \frac{p(i)}{\|\mathbf{p}\|_2} \frac{q(j)}{\|\mathbf{q}\|_2}$. Because the refinement matrix \mathbf{R} reflects closeness of the terms, $\hat{R}(i) \otimes \hat{R}(j)$ would be close to 1 for terms i and j that belong to the same cluster and close to 0 otherwise. Therefore, the above similarity measure is close in value to $\sum_i \sum_j \frac{p(i)}{\|\mathbf{p}\|_2} \frac{q(j)}{\|\mathbf{q}\|_2} = \frac{xy}{\|\mathbf{p}\|_2 \|\mathbf{q}\|_2}$.

If $\|\mathbf{p}\|_2 \cdot \|\mathbf{q}\|_2 = \sqrt{(\sum q(i))^2 (\sum p(i))^2}$, we get the true similarity measure. However, because $\|\mathbf{p}\|_2 \|\mathbf{q}\|_2 = \sqrt{(\sum q(i)^2) (\sum p(i)^2)} \leq (\sum q(i)) (\sum p(i))$, the refinement may overestimate the similarity. On the other hand, noise in data may cause $R(i, j)$ to be smaller than 1. Therefore we may underestimate the similarity. In general, the two types of errors tend to cancel each other, because in situations where overestimation is more likely to happen so would underestimation be. Our experiments on real data show that the refined measure approximates the true similarity well as it does lead to better clustering.

2.2 Similarity Refinement for Set Representation

When we represent the objects of one type by a set of objects of the other type, the similarity between the objects can be measured by normalized set intersections. In this case, the data set can often be modeled by a bipartite graph $G = (A, B, E)$. The nodes in A correspond to one type of object and the nodes in B correspond to another. The edges between the nodes in A and the nodes in B represent the relationship between the objects. (We give an example bipartite graph in panel (1) of Fig. 2.)

For a node $a \in A$, let $\Gamma(a) \subseteq B$ be the neighborhood of a in B . The set $\Gamma(a)$ is then used to represent the node a . The similarity between two nodes a_1 and a_2 is defined to be $\frac{|\Gamma(a_1) \cap \Gamma(a_2)|}{|\Gamma(a_1) \cup \Gamma(a_2)|}$ where $|\cdot|$ means the size of the set. For example, in panel (1) of Fig. 2, a_1 is represented by $\{b_1, b_2\}$ and a_2 by $\{b_1, b_2\}$. The similarity between a_1 and a_2 is then 1. (We use an unweighted bipartite graph and use set operations to measure similarity here. However, the measurement as well as our refinement can be easily extended to weighted bipartite graphs and multisets.)

Before we present our similarity refinement for this representation, we give a graph interpretation of the normalized set intersection. Our similarity refinement is based on this interpretation, which casts the set intersection problem as a maximum flow problem. To illustrate this, we expand the bipartite graph in panel (1) of Fig. 2 by adding a set of nodes A' that mirrors A and a set of nodes B' that mirrors B . The edges between A' and B' also mirror the edges between A and B . Because $b' \in B'$ is the mirror of $b \in B$, we put an edge (b, b') between every such pair of nodes. We then give directions to the edges. The edges between A and B all go from A to B . The edges between B and B' go from B

to B' , and the edges between B' and A' go from B' to A' . (Part of the graph resulting from such expansion is given in panel (2) of Fig. 2. To simplify the figure, we show only nodes and edges that are pertinent to a_1, a_3 , and a_4 .)

We observe that the size of $\Gamma(a_i) \cap \Gamma(a_j)$ in the original graph is equal to the maximum flow going from a_i to a'_j in the expanded graph. For example, a_1 shares one single neighbor with a_3 in the original graph. The flow from a_1 to a'_3 has also maximum size one. a_3 and a_4 share no common neighbors, so the maximum flow from a_3 to a'_4 is zero.

Under this interpretation, there is a natural way to refine the similarity: if two objects b_i and b_j belong to the same cluster, we add an edge between b_i and b'_j such that a bigger flow can get across. In particular, we view the refinement matrix as the adjacent matrix of the bipartite graph on $B \cup B'$. We add an edge (b_i, b'_j) if the entry $R_B(i, j)$ in the refinement matrix \mathbf{R}_B is not zero. The edge is weighed by the value of $R_B(i, j)$. To summarize, let $G(A, B, \mathbf{R}_B)$ be the graph constructed in the above way and let $f = \text{max_flow}(G, a_i, a'_j)$ be the maximum flow in this graph from source a_i to sink a'_j , then $\text{sim}(a_i, a_j, \mathbf{R}_B) = f / (|\Gamma(a_i)| + |\Gamma(a_j)| - f)$.

In the example in Fig. 2, panel (3) shows the result of refining the graph in panel (2). (In the spectral-embedding space, b_1 and b_2 are aligned and they are orthogonal to b_3 and b_4 . This gives a refinement matrix that only adds edges between b_1, b'_2 (b_2, b'_1) and edges between b_3, b'_4 (b_4, b'_3 .) The flow size from a_3 to a'_4 now becomes two and it makes the similarity value between a_3 and a_4 one.

We remark that although flow computation can be expensive, for many real problems, the subgraph involved in estimating the similarity between a_i and a_j can be extremely small. For these problems, our set similarity refinement will have a reasonable complexity.

3 Experiments

3.1 Data Sets and Experiments Setup

We test our co-clustering schemes on real and synthesis data. The similarity refinement for set representation is tested on synthesized bipartite graphs. The similarity refinement for vector representation is tested on the real data sets produced from the 20-News group data [3].

The synthesized bipartite graphs are unweighted. Each graph consists of two sets of nodes U and V , and the nodes in U and V are further divided into clusters, i.e., $U = \cup U_i$ and $V = \cup V_j$. The edges between U_i and V_j are generated according to Bernoulli distribution. The parameters for the distribution that produces the whole graph form a matrix \mathbf{P} such that the (i, j) entry of the matrix is the parameter used to generate the edges from the cluster U_i to the cluster V_j . We summarize the parameters in Table 1.

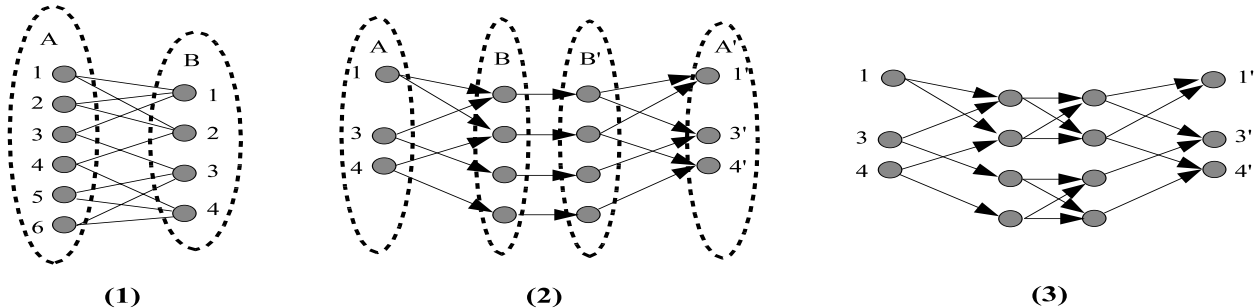


Figure 2. Similarity Refinement for Set Representation

Data Set	Included Newsgroups	# of Docs per Group	Total # Docs
NG1	rec.sport.baseball, rec.sport.hockey	200	400
NG2	comp.os.ms-windows.misc, comp.windows.x, rec.motorcycles, sci.crypt, sci.space	200	1000
NG3	comp.os.ms-windows.misc, comp.windows.x, misc.forsale, rec.motorcycles, sci.crypt, sci.space, talk.politics.mideast, talk.religion.misc	200	1600

Table 2. Newsgroup Testing Data Sets

Data Set	G1	G2
P	$\begin{bmatrix} 0.4 & 0.7 \\ 0.5 & 0.6 \end{bmatrix}$	0.4 0.7
		0.5 0.6
		0.8 0.6

Table 1. Parameters for Generating Bipartite Graphs

The real data sets are obtained from the 20-Newsgroup data set that contains about 20,000 articles from 20 newsgroups. The 20-Newsgroup data set is commonly used for testing document clustering. Our data sets consist of various subsets of the 20-Newsgroup data. The same subsets have been used in previous studies of co-clustering [1, 4]. We construct the document-word matrices using the Rainbow tools [6]. Rainbow tools provide default preprocessing that includes removing stop words and removing headers of the newsgroup articles. (The headers contain the titles of the newsgroups to which the articles belong.) We also select the top 2000 words by mutual information. The document-word matrices are then constructed using the $tf \times idf$ scheme, and each document vector is normalized to have unit 2-norm. We then construct the testing data sets using a subset of the newsgroups and randomly sample 200 articles from each of the newsgroups. The subsets of newsgroups and the numbers of articles in the data sets are summarized in Table 2.

For the synthesized data, we know the number of clusters. We use this number in our experiments. For the real data sets derived from the 20-Newsgroup data, we know the

number of clusters for the documents but do not know the number of clusters for the words. We simply set the word cluster number to be 15.

The true clusters in the 20-Newsgroup data are known. For the synthesized bipartite graph, the clusters we specified when generating the graph may not be the best clustering for the graph. For example, consider the bipartite graph G1. Conceptually, cluster U_1 should consist of the nodes whose neighborhood in V_1 is smaller than its neighborhood in V_2 . The ratio of the two neighborhoods should be around $4/7$. Cluster U_2 should consist of the nodes for which that ratio is about $5/6$. Because we generate the edges in a random way, there is a small chance that a node (say u_i) in U_1 may achieve a ratio close to, or even larger than $5/6$. And a node (say u_j) in U_2 may give a ratio smaller than $4/7$. As there is no other information but the ratio by which we determine the clustering of the nodes, u_i should be put in cluster U_2 while u_j should be put in cluster U_1 . In our experiments, we use such modified clusters as the true clustering. For example, for G1, the true clustering would put in one cluster the nodes whose V_1 to V_2 neighborhood-size ratio is close to $4/7$ and in another cluster the nodes whose ratio is close to $5/6$.

We use the normalized mutual information (NMI) between the clustering result and the true clusters to measure the quality of the clustering. For bipartite graphs, we use the NMI of the clustering of U . For the 20-Newsgroup data, we use the NMI of the document clustering. Each experiment is repeated five times to obtain the NMI scores. (In each repetition, we regenerate the graphs and resample the doc-

uments. Therefore, the variance of the scores includes the variance of the clustering performed on different samples.)

3.2 Results and Discussion

When presenting the experiment results, we refer to our co-clustering scheme as SRCC (similarity refinement co-clustering). For the newsgroup data, we compare our co-clustering to the results of [4]. (Our newsgroup testing data sets are constructed exactly the same way as the ones in [4]. Therefore, we use the results directly from [4] for comparison.) We refer to the clustering scheme in [4] as RSN. Several variations of RSN clustering were introduced in [4]. When comparing the results, we always compare to the variation of RSN that performs the best. For the synthesized bipartite data, we compare our co-clustering to the standard kmeans clustering, which we refer to as KM.

Data Set	SRCC	RSN
NG1	0.901 ± 0.043	0.638 ± 0.164
NG2	0.807 ± 0.022	0.747 ± 0.068
NG3	0.749 ± 0.013	0.698 ± 0.037

Table 3. NMI of Newsgroup Data Clustering

Table 3 shows the clustering results for the newsgroup testing data. We see that for all the data sets, SRCC performs better than RSN. In particular, SRCC performs very well for the data set NG1. NG1 contains two close newsgroups. It is a challenging data set for the RSN clustering scheme. On average, our clustering achieves an NMI score over 0.9 for this data set. Out of the 400 documents in NG1, only fewer than five documents were put in the wrong clusters by SRCC. From NG1 to NG3, there are more documents and clusters in the data set. It is expected that performance may go down while the problem becomes more complex. The NMI scores of SRCC show such a trend. However, SRCC performance is always better than RSN for all data sets. It indicates that after refinement, the resulting similarity measure does provide a good approximation to the true similarities among the documents. Therefore, refinement leads to clustering closer to the true clusters.

Data Set	SRCC	KM
G1	0.786 ± 0.038	0.238 ± 0.052
G2	0.822 ± 0.019	0.680 ± 0.011

Table 4. NMI of Bipartite Graph Clustering

Table 4 summarizes the results for the bipartite graph clustering. Again, the results indicate that refinement leads to better clustering. In particular, testing data G1 is a challenging case for kmeans, as the parameters used in generating edges for different clusters are quite similar. However,

our similarity refinement still produces clustering much closer to the true clusters.

4 Conclusion

We have introduced a similarity refinement framework for co-clustering. Given a data set that contains objects of different types, when clustering objects of one type, it is often necessary to consider the cluster structures on the objects of the other types. Our framework uses a spectral-embedding-based approach to obtain an approximate cluster structure on objects of one type and then uses this information to refine the similarity measures for the objects of other types. Our experiments show that the refinement leads to better clustering, indicating that refined similarity is closer to the true similarity.

References

- [1] I. S. Dhillon, S. Mallela, and D. S. Modha. Information-theoretic co-clustering. In *Proceedings of The Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 89–98, 2003.
- [2] Chris H. Q. Ding, Tao Li, Wei Peng, and Haesun Park. Orthogonal nonnegative matrix t-factorizations for clustering. In *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 126–135, 2006.
- [3] Ken Lang. NewsWeeder: learning to filter netnews. In *Proceedings of the 12th International Conference on Machine Learning*, pages 331–339, 1995.
- [4] Bo Long, Xiaoyun Wu, Zhongfei (Mark) Zhang, and Philip S. Yu. Unsupervised learning on k-partite graphs. In *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 317–326, 2006.
- [5] Bo Long, Zhongfei (Mark) Zhang, and Philip S. Yu. Co-clustering by block value decomposition. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 635–640, 2005.
- [6] Andrew Kachites McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. <http://www.cs.cmu.edu/~mccallum/bow>, 1996.
- [7] Andrew Y. Ng, Michael Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems*, 2001.
- [8] Suvrit Sra, Hyuk Cho, Inderjit S. Dhillon, and Yuqiang Guan. Minimum sum-squared residue co-clustering of gene expression data. In *Proceedings of the Fourth SIAM International Conference on Data Mining*, 2004.
- [9] Hongyuan Zha, Xiaofeng He, Chris H. Q. Ding, Ming Gu, and Horst D. Simon. Bipartite graph partitioning and data clustering. In *CIKM*, pages 25–32, 2001.