

# Preserving Patterns in Bipartite Graph Partitioning

Tianming Hu

DongGuan U. of Technology  
tmhu05@gmail.com

Chao Qu

DongGuan U. of Technology  
chaost@dgut.edu.cn

Chew Lim Tan

National U. of Singapore  
tancl@comp.nus.edu.sg

Sam Yuan Sung

South Texas College  
sysung@southtexascollege.edu

Wenjun Zhou

Rutgers University  
wjzhou@pegasus.rutgers.edu

## Abstract

*This paper describes a new bipartite formulation for word-document co-clustering such that hyperclique patterns, strongly affiliated documents in this case, are guaranteed not to be split into different clusters. Our approach for pattern preserving clustering consists of three steps: mine maximal hyperclique patterns, form the bipartite, and partition it. With hyperclique patterns of documents preserved, the topic of each cluster can be represented by both the top words from that cluster and the documents in the patterns, which are expected to be more compact and representative than those in the standard bipartite formulation. Experiments with real-world datasets show that, with hyperclique patterns as starting points, we can improve the clustering results in terms of various external clustering criteria. Also, the partitioned bipartite with preserved topical sets of documents naturally lends itself to different functions in search engines.*

## 1. Introduction

Document clustering is usually based upon their word distributions, while word clustering is determined by co-occurrence in documents. Such a duality between document and word clustering can be modeled with a bipartite, where documents and words are modeled as vertices on two sides respectively, and only edges linking different kinds of vertices are considered [7]. Finding an optimal partitioning in such a bipartite gives a co-clustering of documents and words. It is expected that top documents and words in the same cluster can represent its topic, where top vertices usually means the ones with highest degrees within the cluster.

However, such claims may fail if the cluster is not

pure enough or it includes words/documents with very general topics. To perform natural clustering and to precisely capture the cluster topic, we need to identify those micro-sets of words/documents that are very similar and typical of their respective topics. Besides, we need to ensure that they would not be separated into different clusters during the clustering process.

Recently, a new pattern for association analysis[2], hyperclique pattern, was defined in [30]. Hyperclique patterns truly possess such desirable property: the objects in a hyperclique pattern have a guaranteed level of global pairwise similarity to one another as measured by the cosine or Jaccard similarity measures [31]. Since clustering depends on similarity, it is expected that good clustering algorithms should not break the hyperclique pattern. However, this is not the case for traditional clustering algorithms, as demonstrated in [29]. There are two main reasons: 1)clustering algorithms have no built-in knowledge of these patterns; 2)many clustering techniques produce a partitioning of disjoint clusters, while hyperclique patterns are sometimes overlapping.

By now, a natural question arises if we can use hyperclique patterns explicitly such that they are preserved in the resultant clustering. A clustering technique automatically preserve patterns if it has the following two properties. First, it can be set such that clustering begins with starting points that are either original objects or patterns. Second, it must not breakup these points of patterns during the clustering process. According to these two properties, this paper proposes a new bipartite formulation for word-document co-clustering, where hyperclique patterns of documents are guaranteed not to be split into different clusters. Our approach, BiPartite GrAph partitioning with Pattern preservation (BIGAP), is compared with the standard bipartite formulation on five real datasets,

using four external evaluation criteria. The empirical results show that we can make improvement in terms of all criteria.

The Hierarchical Clustering with Pattern Preservation (HICAP) algorithm [29] also used hyperclique patterns as starting points. The major difference is that HICAP is purely on clustering documents, which is very different from our case, co-clustering words and documents in the bipartite. Besides, the high computational cost of hierarchical clustering practically prevents its use in very large datasets, e.g., for search engines, which is elaborated a little bit below.

Due to the high affiliation within patterns, the pattern preserving partitioned bipartite naturally lends itself to various functions in search engines. For instance, instead of a long ranked list for keyword queries, it is better to return clustered search results by topic. This can be done by showing only the documents in the patterns, which must be more compact and representative of those topics. For a query session, instead of returning the documents where all the query words just co-occur, we can select to return the documents from hyperclique patterns that connect the queries and embody their common topic. Because we also have a by-product of word clustering in the meantime, it can be used for relevant term suggestion such as search engine advertisement words.

The rest of the paper is organized as follows. Section 2 describes related work and Section 3 presents the details of our algorithm. Experimental results are reported in Section 4, together with a demonstration on its applications to search engines. Finally we conclude this paper in Section 5.

## 2. Background and related work

### 2.1. Document clustering

Clustering has been extensively studied in data mining, statistics and pattern recognition fields [15]. Clustering algorithms generally come in two categories: partitional and hierarchical. The former produces a set of un-nested clusters by partitioning the data into disjoint groups. The latter produces a nested sequence of partitionings, with a single cluster at the top and singleton clusters at the bottom. Approaches to document clustering can be also roughly divided into these two classes.

Hierarchical clustering approaches falls into two classes: divisive and agglomerative. Group Average (UPGMA) belongs to the latter and defines cluster similarity in terms of the average pairwise similarity between the points in the two clusters. A recent study

[33] found UPGMA to be the best in this class for clustering text. As for the partitional clustering, probably K-means is the most widely used method. As a modification, bisecting K-means [22] can also be employed in hierarchical clustering of documents and produces competitive results [22, 33].

### 2.2. Graph based document clustering

Graph-theoretic techniques have also been considered for clustering[23]. They model the document similarity by a graph whose vertices correspond to documents and weighted edges give the similarity between vertices. Graphs can also model words as vertices and similarity between words is based on documents in which they co-occur. Partitioning the graph yields a clustering of words, which is assumed to be associated with similar concepts[4].

Document clustering is based upon their word distributions, while word clustering is determined by co-occurrence in documents. Such a duality between document and word clustering can be modeled using a bipartite, where documents and words are modeled as vertices on two sides respectively, and only edges linking different kinds of vertices are considered [7]. Finding an optimal partitioning in such a bipartite gives a co-clustering of documents and words, with the expectation that documents and words in the same cluster are related to the same topic.

### 2.3. Pattern preserving clustering

Pattern preserving clustering is related to constrained clustering and frequent item set based clustering. Constrained clustering [25] is based on standard clustering approaches with additional restriction on the clustering process. Hyperclique pattern preserving clustering can be viewed as constraining certain objects to stay together during the clustering process. Such constraints are automatically imposed in our case of graph partitioning with patterns as vertices. There have been other clustering approaches based on frequent itemsets [5, 19, 6]. They are quite different from ours, for we start directly with sets of correlated objects which may not be frequent. Besides, they are not pattern preserving.

Since agglomerative clustering approaches starts with individual objects as clusters, and then successively combining the two most similar clusters, they automatically preserve patterns if patterns are used as starting points. Based on this observation, [29] proposed HICAP and showed that its clusters are more interpretable than those of UPGMA. HICAP is purely

on clustering documents, which is very different from our case, co-clustering words and documents in the bipartite. Besides, the high cost of hierarchical clustering practically prevents its use in large datasets, e.g., for search engines.

## 2.4. Relations to search engines

In this subsection, we briefly review related work on search engines, where pattern preserving partitioned bipartites can play a role.

For keyword queries, current search engines normally return a long ranked list of several hundred documents and leave it to the user to find which ones are of his interest. Users, on the other hand, usually only explore the first one or two pages. Because high ranked documents may not meet the user need, it is better to give the user a quick view of the whole results, say, by returning clustered search results by topic. Related work includes the topic sensitive PageRank that computes a set of PageRank vectors biased using a set of representative topics [12]. Vivisimo[26] provides clustered search results based on distinct frequent words. Within the cluster, the documents are still shown according to their original ranks. However, a frequent word may not represent a topic and it may be meaningless. Here for each topic(cluster), we can select to show only the documents in the patterns and use them for generating topical words.

Search engines often provide query term suggestions, which attempts to suggest relevant terms to help users formulate more effective queries. Document-based approaches extract co-occurring key terms from retrieved documents that are ranked high [32]. Log-based approaches identify relevant query terms in collected logs of user queries, e.g., by clustering queries based on ‘click-through data’ composed of the query and the URLs that the user actually visits among the list provided by the search engine [28]. The main problem with the above two kinds of approaches is that they only use high-ranked search results, which may not be relevant to the topic and may cause bias to the clustering results. Query session-based approaches suggest for a user query those that co-occur in similar query sessions from search engine logs, totally ignoring the retrieved documents [13]. With pattern preserving bipartites, we can make use of contextual information from both queries and retrieved documents.

## 2.5. Hyperclique patterns

In this paper, hyperclique patterns are what we preserve during clustering. They are based on the concepts

on frequent itemsets. In this subsection, we first briefly review the concepts on frequent itemsets, then describe the concept of hyperclique patterns.

Let  $I = \{i_1, i_2, \dots, i_m\}$  be a set of distinct items. Each transaction  $T$  in database  $D$  is a subset of  $I$ . We call  $X \subseteq I$  an itemset. An itemset with  $k$  items is called a  $k$ -itemset. The support of  $X$ , denoted by  $supp(X)$ , is the fraction of transactions containing  $X$ . If  $supp(X)$  is no less than a user-specified minimum support,  $X$  is called a frequent itemset. The confidence of association rule  $X_1 \rightarrow X_2$  is defined as  $conf(X_1 \rightarrow X_2) = supp(X_1 \cup X_2)/supp(X_1)$ . It estimates the likelihood that the presence of a subset  $X_1 \subseteq X$  implies the presence of the other items  $X_2 = X - X_1$  in the same transaction.

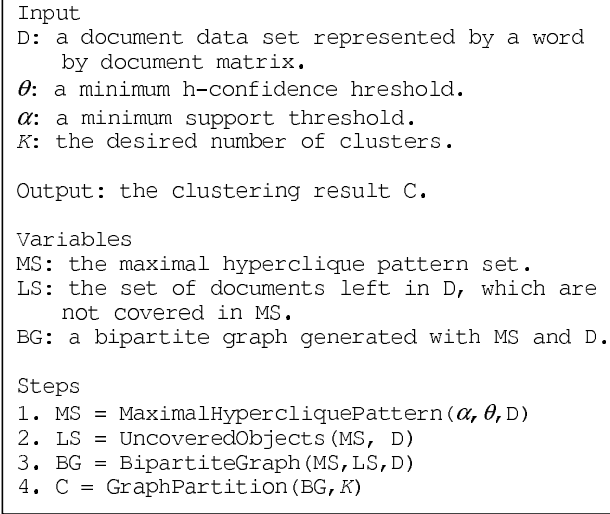
If the minimum support threshold is low, we may extract too many spurious patterns involving items with substantially different support levels, such as (caviar, milk) in the basket data. If the minimum support threshold is high, we may miss many interesting patterns occurring at low levels of support, such as (caviar, vodka). To measure the overall affinity among items within an itemset, the h-confidence was proposed in [30]. Formally, the h-confidence of an itemset  $P = \{i_1, i_2, \dots, i_m\}$  is defined as  $hconf(P) = \min_k \{conf(i_k \rightarrow P - i_k)\}$ . Given a set of items  $I$  and a minimum h-confidence threshold  $h_c$ , an itemset  $P \subseteq I$  is a hyperclique pattern if and only if  $hconf(P) \geq h_c$ . A hyperclique pattern  $P$  can be interpreted as that the presence of any item  $i \in P$  in a transaction implies the presence of all other items  $P - \{i\}$  in the same transaction with probability at least  $h_c$ . This suggests that h-confidence is useful for capturing patterns containing items which are strongly related with each other. A hyperclique pattern is a maximal hyperclique pattern if no superset of this pattern is a hyperclique pattern.

## 3. Bipartite graph partitioning with pattern preservation

BIGAP is based on the bipartite graph partitioning with hyperclique patterns as vertices. So the objects in the hyperclique pattern will not be separated during graph partitioning. Fig. 1 gives the overview of the algorithm. Detailed description is given later in this section.

### 3.1. Mining maximal hyperclique patterns

To apply clustering algorithms, a document data set is usually represented by a matrix. First we extract from documents unique content-bearing words as features, which involves removing stopwords and those



**Figure 1. Overview of the BIGAP algorithm.**

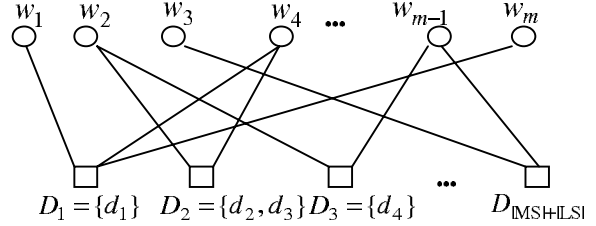
with extreme document frequencies. More sophisticated techniques use support or entropy to filter words further. Then each document is represented as a vector in this feature space. With rows for words and columns for documents, the word by document matrix  $A$ 's non-zero entry  $A_{ij}$  indicates the presence of word  $w_i$  in document  $d_j$ , while a zero entry indicates an absence.

By regarding words as transactions and documents as items, we first mine all maximal hyperclique patterns of documents, which are the patterns we want to preserve. For this purpose, we employ a hybrid approach proposed in [14], which exploited key advantages of both the depth first search strategy and the breadth first search strategy. The experimental results showed that it can be orders of magnitude faster than standard maximal frequent pattern mining algorithms, particularly at low levels of support.

### 3.2. Forming the bipartite

First some notations for general graph representation. A graph  $G = (V, E)$  is composed of a vertex set  $V = \{1, 2, \dots, |V|\}$  and an edge set  $\{(i, j)\}$  each with edge weight  $E_{ij}$ . The graph can be stored in an adjacency matrix  $M$ , with entry  $M_{ij} = E_{ij}$  if there is an edge  $(i, j)$ ,  $M_{ij} = 0$  otherwise.

Given the  $m \times n$  word-by-document matrix  $A$ , the standard bipartite graph  $G = (V, E)$  is constructed as follows. First we order the vertices such that the first  $m$  vertices index the words while the last  $n$  index the documents, so  $V = V_W \cup V_D$ , where  $V_W$  contains  $m$  vertices each for a word, and  $V_D$  contains  $n$  vertices each



**Figure 2. The bipartite with meta-documents.**

for a document. Edge set  $E$  only contains edges linking different kinds of vertices, so the adjacency matrix  $M$  may be written as  $\begin{pmatrix} 0, A \\ A^T, 0 \end{pmatrix}$ .

In our case, with the maximal hyperclique patterns of documents MS, we first identify those remaining documents LS that never appear in MS. Then we construct vertex set  $V = V_W \cup V_D$  as follows.  $V_W$  still contains  $m$  vertices each for a word. As shown in Fig. 2,  $V_D$  contains  $|MS| + |LS|$  vertices each for a meta-document, i.e., either a pattern in MS or a document in LS. The new  $m \times (|MS| + |LS|)$  word by meta-document matrix  $A'$  is defined as  $A'_{ij} = \sum_{d_k \in D_j} A_{ik}$ . That is, the association between word  $w_i$  and meta-document  $D_j$  is the sum of association between word  $w_i$  and all documents  $d_k$  in  $D_j$ .

### 3.3. Graph partitioning

Given a weighted graph  $G = \{V, E\}$  with adjacency matrix  $M$ , clustering the graph into  $K$  parts means partitioning  $V$  into  $K$  disjoint clusters of vertices  $V_1, V_2, \dots, V_K$ , by cutting the edges linking vertices in different parts. The general goal is to minimize the sum of the weights of those cut edges. Formally, the cut between two vertex groups  $V_1$  and  $V_2$  is defined as  $cut(V_1, V_2) = \sum_{i \in V_1, j \in V_2} M_{ij}$ . Thus the goal can be expressed as  $min_{\{V_1, V_2, \dots, V_K\}} \sum_{k=1}^K cut(V_k, V - V_k)$ . To avoid trivial partitions, often the constraint is imposed that each part should be roughly balanced in terms of part weight  $wgt(V_k)$ , which is often defined as sum of its vertex weight. That is,  $wgt(V_k) = \sum_{i \in V_k} wgt(i)$ . The objective function to minimize becomes  $\sum_{k=1}^K cut(V_k, V - V_k) / wgt(V_k)$ . Given two different partitionings with the same cut value, the above objective function value is smaller for the more balanced partitioning.

In practice, different optimization criteria have been defined with different vertex weights. The ratio cut criterion [10], used for circuit partitioning, defines  $wgt(i) = 1$  for all vertices  $i$  and favors equal sized clusters. The normalized cut criterion [21], used for image

segmentation, defines  $wgt(i) = \sum_j M_{ij}$ . It favors clusters with equal sums of vertex degrees, where vertex degree refers to the sum of weights of edges incident on it.

Finding a globally optimal solution to such a graph partitioning problem is in general NP-complete [9], though different approaches have been developed for good solutions in practice [16, 1, 8]. Here we employ Graclus [8], a fast kernel based multilevel algorithm, which involves coarsening, initial partitioning and refinement phases. Unlike existing multilevel approaches like METIS [16], Graclus does not constrain the cluster sizes to be nearly equal. Recently graph partitioning with a general cut objective was shown to be mathematically equivalent to an appropriate weighted kernel K-means objective function. Graclus exploits this equivalence to perform K-means at multilevels during refinement to ensure decrease in the objective function further.

## 4. Experimental evaluation

In this section, we present an experimental evaluation of BIGAP. First we introduce the experimental datasets and cluster evaluation criteria, then we evaluate the clustering performance of BIGAP against the standard bipartite formulation.

### 4.1. Experimental datasets

For evaluation, we used five real data sets from different domains. The LA1 data set is part of the TREC-5 collection [24] and contains news articles from the Los Angeles Times. The RE0 and RE1 data sets are from the Reuters-21578 text categorization test collection Distribution 1.0 [17]. The data set WAP is from the WebACE project [11]; each document corresponds to a web page listed in the subject hierarchy of Yahoo. The data set TR31 was derived from the TREC-5 collection [24]. For all data sets, we used a stoplist to remove common words, and stemmed the remaining words using Porter’s suffix-stripping algorithm [20]. Some characteristics of these data sets are shown in Table 1.

### 4.2. Evaluation criteria

Because the true class labels of documents are known, we can measure the quality of the clustering solutions using external criteria that measure the discrepancy between the structure defined by a clustering and what is defined by the class labels. First we compute the confusion matrix  $C$  with entry  $C_{ij}$  as the number of

**Table 1. Characteristics of data sets.**

data	LA1	RE0	RE1	WAP	TR31
#doc	3204	1504	1657	1560	927
#word	31472	2886	3758	8460	10128
#class	6	13	25	20	7
MinClass	273	11	13	5	2
MaxClass	943	608	371	341	352
min/max	0.29	0.018	0.035	0.015	0.006

documents from true class  $j$  that are assigned to cluster  $i$ . Then we calculate the following four measures: normalized mutual information (NMI), conditional entropy (CE), error rate (ERR) and F-measure.

NMI and CE are entropy based measures. The cluster label can be regarded as a random variable with the probability interpreted as the fraction of data in that cluster. Let  $T$  and  $C$  denote the random variables corresponding to the true class and the cluster label, respectively. The two entropy-based measures are computed defined as  $NMI = (H(T) + H(C) - H(T, C)) / \sqrt{H(T)H(C)}$  and  $CE = H(T|C) = H(T, C) - H(C)$ , where  $H(X)$  denotes the entropy of  $X$ . NMI measures the shared information between  $T$  and  $C$ . CE tells the information remained in  $T$  after knowing  $C$ . Error rate  $ERR(T|C)$  computes the fraction of misclassified data when all data in each cluster is classified as the majority class in that cluster. It can be regarded as a simplified version of  $H(T|C)$ .

F-measure combines the precision and recall concepts from information retrieval [3]. We treat each cluster as if it were the result of a query and each class as if it were the desired set of documents for a query. We then calculate the recall and precision of that cluster for each given class as follows:  $R_{ij} = C_{ij}/C_{+j}$ ,  $P_{ij} = C_{ij}/C_{i+}$ , where  $C_{+j}/C_{i+}$  is the sum of  $j$ th column/ $i$ -th row, i.e.,  $j$ -th class size / $i$ -th cluster size. Note that  $C_{+j}$  could be larger than the true size of class  $j$  if some documents from it appear in more than one cluster. F-measure of cluster  $i$  and class  $j$  is then given by  $F_{ij} = 2R_{ij}P_{ij} / (P_{ij} + R_{ij})$ . The overall value for the F-measure is a weighted average for each class:  $F = \sum_j C_{+j} \max_i \{F_{ij}\} / n$ , where  $n$  is the total sum of all elements of matrix  $C$ . F-measure reaches its maximal value of 1 when the clustering is the same as the true classification.

### 4.3 Hyperclique pattern of words

Before presenting clustering results, let us have a flavor of words in the hyperclique pattern. By treating

**Table 2. Example patterns of words from WAP.**

pattern	support	h-confidence
solaris, unix, sparc	0.19%	100%
wnba, lineman	0.19%	75%
martina, hingis	0.51%	80%
wavetop, wavephore, tuner	0.51%	75%

documents as transactions and words as items, Table 2 shows example word patterns from WAP, whose documents are from categories like people, television, etc. Apparently, the first two rows are from class technology and sports, respectively. Martina Hingis is a star player of woman tennis. WaveTop is a data broadcast software by WavePhor Inc. that allows users with TV tuner-enabled PCs to watch cable TV. Although their h-confidence is high, their support is low, which means there are few edges between them and the documents. Thus it is very possible for graph partitioning algorithms to assign words of a pattern into different clusters. Taking them as initial clusters, however, prevents them from being separated.

#### 4.4 Clustering results

All the datasets used here are of transactional form, that is, the word by document matrix  $A$  is binary. Although the hybrid approach [14] used to mine hyperclique patterns can only handle transactional data, we could use other values such as term frequency for entry  $A_{ij}$  during the graph generation phase. This is expected to produce better clustering results. Nevertheless, because our main purpose is to show the advantage of using hyperclique patterns as starting points, we just compare BIGAP with the standard bipartite formulation on transactional data.

As for the graph partitioning criterion used in Graculus, we choose the normalized cut criterion instead of the ratio cut criterion for two reasons. First, as shown above, our datasets are highly imbalanced, which makes unreasonable the constraint of equal sized clusters by the ratio cut criterion. Second we find that sometimes it yields clusters of pure word vertices, which makes it impossible to determine the number of document clusters (clusters containing the document vertices) beforehand. Those words with low frequencies are likely to be isolated together, since few edges linking outside are cut. As for the normalized cut criterion that tries to balance sums of vertex degrees in each cluster, the resultant clusters tend to contain both

**Table 3. Comparison on five datasets.**

data	method	ERR	F	NMI	CE
LA1	BIGAP	0.5379	0.4734	0.2605	0.4844
	STD	0.7057	0.3045	0.1012	1.1256
RE0	BIGAP	0.4152	0.3675	0.3304	1.6739
	STD	0.4501	0.3024	0.2698	1.8858
RE1	BIGAP	0.5202	0.3445	0.3628	1.7389
	STD	0.5890	0.3052	0.3139	1.6068
WAP	BIGAP	0.4795	0.4579	0.4728	0.9269
	STD	0.5179	0.3640	0.3895	1.1671
TR31	BIGAP	0.4276	0.5816	0.2715	0.7870
	STD	0.4984	0.5344	0.2644	0.6253

**Table 4. Top words from WAP clusters.**

subject	words
film	hollywood, star, film, director, comedy
sports	champion, finish, brian, tournament, jordan
TV	screen, night, cable, tell, feel
politics	washington, republican, senator, white, campaign
business	stock, financial, quote, wire, price
technology	internet, computer, access, technology, right
people	diana, princess, paris, crash, car
health	test, discover, diet, process, gene

document and word vertices.

By setting the number of clusters equal to the true number of classes, the clustering results are shown in Table 3, where STD denotes the standard bipartite formulation. NMI and F are preferred large while ERR and CE are preferred small. One can see that BIGAP is able to achieve improvement on all five datasets in terms of nearly all four measures. The only exception is that BIGAP leads to worse CE values on RE1 and TR31. The two parameters, support threshold, h-confidence threshold, were tuned separately for each dataset, but not for each criterion. For instance, all results on RE1 in the table are recorded at ( $supp = 0.005$ ,  $hconf = 0.5$ ), though better CE results could be obtained at other parameter settings.

Ideally, when obtained clustering is the same as true classification, every row(cluster) in the confusion matrix should contain exactly one nonzero value. Although the word by document matrix used here is binary, the results on some datasets are promising, as indicated by many zeros in their confusion matrices. Fig. 3 plots the confusion matrices for WAP and RE0. Each matrix is first normalized by dividing it with the maximum entry and then entries are shown by darkness proportional to their values. One can see there are many white (zero) entries and a number of rows

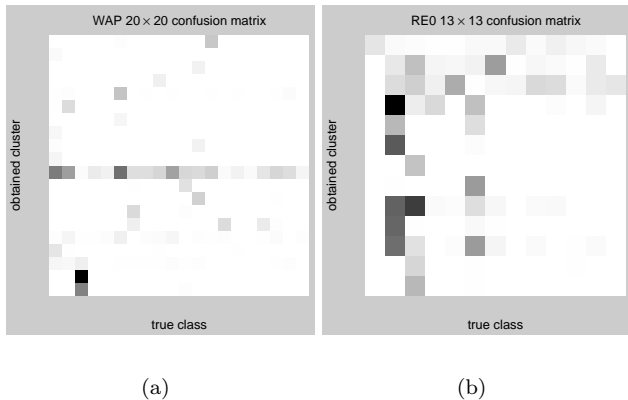


Figure 3. Confusion matrices for WAP and RE0.

have only one nonzero entry. Table 4 illustrates the top five words from some of clusters for WAP, which are obviously representative of corresponding subjects.

**Discussions.** The clusters corresponding to the last two rows in Table 4 are both nearly pure. At first glance, words like paris, crash and car have nothing to do with subject people, whose top words are supposed to be something like names and titles. However, they are really dominant words in that cluster, due to the extensive reports of tragedy of Princess Diana. For the itemset  $I = \{diana, princess, paris, crash, car\}$ , its two parameters are  $supp = 0.025$ ,  $hconf = 0.29$ . Since words like princess/paris are too general, the documents it appears can talk about any topic, so the h-confidence for itemset  $\{princess, paris\}$  is generally low, not to mention  $\{princess, I - princess\}$ . Nevertheless, if we generalize item to superitem (i.e., itemset), the h-confidence is about 1 for 2-superitem-set  $\{(diana, car), (princess, paris, crash)\}$ . Mining such hyperclique patterns of superitems deserves further investigation.

#### 4.5 Applications to search engines

In this subsection, we illustrate potential applications of such pattern preserving partitioned bipartites to search engines. The motivation is still the high affiliation within hypercliques.

Since many words have several topics, it is better to return clustered search results by topic for keyword queries. The standard bipartite formulation can do this job by grouping all the documents containing the query according to the cluster label. In the pattern preserving case, however, we can output only the hyperclique pat-

terns for each cluster, which are often fewer. Besides, they are more representative of the cluster topic, for the documents in the pattern must share many topical words, which is possible only if they are very typical for the topic.

Second, such bipartites can help return more relevant results for a query session. Due to the ambiguity of words and the shortness of queries, it is hard for search engines to return relevant results based on a single query. The user often needs to input more queries in subsequent requests, which forms a query session. It is important to exploit the contextual information in the query session. Current search engines usually just return the documents where the query words co-occur, not the documents of the topic common to them. Given that the user information need is exactly the topic shared by all queries in the session, a better way is to return such documents only from the hyperclique patterns (possibly from multiple clusters), which are more compact and representative. In other words, the hyperclique patterns connect the queries and embody their common topic.

Suppose a user who needs information about Princess Diana’s accident. If he first inputs paris and then princess, besides documents on the accident, he would get irrelevant results from other classes, such as ‘Movie Reviews by Greg Paris - Princess Mononoke’ from class film, and ‘U.S. Women Name Paris Hilton Princess’ from class media. However, if we just return the hyperclique patterns, we have only two documents from a hyperclique pattern in WAP with  $supp = 0.02$ ,  $hconf = 0.82$ , which belongs to a nearly pure cluster corresponding to class people. Both of them also contain words diana, crash and car.

Last such bipartites can be applied to topic sensitive term suggestions. With pattern preserving bipartites, we can make use of contextual information from both queries and retrieved documents. For instance, given all the keywords  $Q$  in the query session so far, we first retrieve the hyperclique patterns of documents where they co-occur from each cluster. Then from the corresponding document patterns of each cluster, we seek key terms  $T$  for which  $\{Q, T\}$  forms a term hyperclique pattern. Such  $T$  is guaranteed to reflect the cluster topic and to be highly affiliated with the query terms.

## 5. Conclusion

In this paper, we presented a new approach, Bipartite GrAPh partitioning with Pattern preservation (BIGAP), for word-document co-clustering. Hyperclique patterns capture strong connections between groups of objects and should not be separated during clustering.

Using them as starting points on the document side in the bipartite, we showed that better clustering results could be obtained in terms of various external criteria in our experiments. Besides, because of high affiliation within the patterns, the cluster topic can be captured more precisely by pattern documents together with the top words in the respective clusters. Due to the unique structure of the partitioned bipartite, it naturally lends itself to various functions in search engines. Finally we illustrated such potential applications for future work.

## References

- [1] K. Andreev and H. Racke. Balanced graph partitioning. In *Proc. 16th Annual ACM Symposium on Parallelism in Algorithms and Architectures*, pages 120–124, 2004.
- [2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. VLDB*, pages 487–499, 1994.
- [3] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, 1999.
- [4] L. D. Baker and A. McCallum. Distributional clustering of words for text classification. In *Proc. SIGIR*, pages 96–103, 1998.
- [5] F. Beil, M. Ester, and X. Xu. Frequent term-based text clustering. In *Proc. SIGKDD*, pages 436–442, 2002.
- [6] M. E. Benjamin, C. M. Fung, and K. Wang. Large hierarchical document clustering using frequent itemsets. In *Proc. SDM*, 2003.
- [7] I. S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proc. SIGKDD*, pages 269–274, 2001.
- [8] I. S. Dhillon, Y. Guan, and B. Kulis. A fast kernel-based multilevel algorithm for graph clustering. In *Proc. SIGKDD*, pages 629–634, 2005.
- [9] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Company, 1979.
- [10] L. Hagen and A. Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE Trans. CAD*, 11:1074–1085, 1992.
- [11] E.-H. Han, D. Boley, M. Gini, R. Gross, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore. Webace: A web agent for document categorization and exploration. In *Proc. Int'l Conf. Autonomous Agents*, pages 408–415, 1998.
- [12] T. H. Haveliwala. Topic-sensitive PageRank: A context-sensitive ranking algorithm for web search. *IEEE Trans. Knowledge and Data Engineering*, 15(4):784–796, 2003.
- [13] C.-K. Huang, L.-F. Chien, and Y.-J. Oyang. Relevant term suggestion in interactive web search based on contextual information in query session logs. *J. American Society for Information Science and Technology*, 54(7):638–649, 2003.
- [14] Y. Huang, H. Xiong, W. Wu, and Z. Zhang. A hybrid approach for mining maximal hyperclique patterns. In *Proc. ICTAI*, pages 354–361, 2004.
- [15] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, 1999.
- [16] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Scientific Computing*, 20(1):359–392, 1998.
- [17] D. Lewis. Reuters-21578 text categorization text collection 1.0. <http://www.research.att.com/~lewis>.
- [18] B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In *Proc. Int'l Conf. Knowledge Discovery and Data Mining*, pages 80–86, 1998.
- [19] J. Pei, X. Zhang, M. Cho, H. Wang, and P. Yu. Maple: A fast algorithm for maximal pattern-based clustering. In *Proc. ICDM*, pages 259–266, 2003.
- [20] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.
- [21] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22:888–905, 2000.
- [22] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. In *KDD Workshop on Text Mining*, 2000.
- [23] A. Strehl, J. Ghosh, and R. Mooney. Impact of similarity measures on web-page clustering. In *Proc. AAAI: Workshop of Artificial Intelligence for Web Search*, pages 58–64, 2000.
- [24] TREC. <http://trec.nist.gov>.
- [25] A. K. H. Tung, R. T. Ng, L. V. S. Lakshmanan, and J. Han. Constraint-based clustering in large databases. In *Proc. ICDT*, pages 405–419, 2001.
- [26] Vivisimo. <http://vivisimo.com/>.
- [27] K. Wang, C. Xu, and B. Liu. Clustering transactions using large items. In *Proc. CIKM*, pages 483–490, 1999.
- [28] J.-R. Wen, J.-Y. Nie, and H.-J. Zhang. Clustering user queries of a search engine. In *Proc. WWW*, pages 162–168, 2001.
- [29] H. Xiong, M. Steinbach, P.-N. Tan, and V. Kumar. HICAP: Hierarchical clustering with pattern preservation. In *Proc. SDM*, 2004.
- [30] H. Xiong, P.-N. Tan, and V. Kumar. Mining strong affinity association patterns in data sets with skewed support distribution. In *Proc. ICDM*, pages 387–394, 2003.
- [31] H. Xiong, P.-N. Tan, and V. Kumar. Hyperclique pattern discovery. In *Data Mining and Knowledge Discovery Journal*, In Press, 2006.
- [32] J. Xu and W. B. Croft. Query expansion using local and global document analysis. In *Proc. SIGIR*, pages 4–11, 1996.
- [33] Y. Zhao and G. Karypis. Evaluation of hierarchical clustering algorithms for document datasets. In *Proc. CIKM*, pages 515–524, 2002.