International Journal on Artificial Intelligence Tools
Vol. 17, No. 1 (2008) 87–107

World Scientific
www.worldscientific.com

CO-CLUSTERING BIPARTITE WITH PATTERN PRESERVATION FOR TOPIC EXTRACTION

TIANMING HU[*,†], CHEW LIM TAN[‡], YONG TANG[§], SAM YUAN SUNG[¶], HUI XIONG[‖] and CHAO QU[†,**]

[*]*School of Software Engineering, East China Normal University*
[†]*Department of Computer Science, Dongguan University of Technology*
[‡]*Department of Computer Science, National University of Singapore*
[§]*Department of Computer Science, Sun Yat-Sen University*
[¶]*Department of Computer Science, South Texas College*
[‖]*Management Science and Information Systems Department, Rutgers University*
[*]*tmhu@ieee.org*
[‡]*tancl@comp.nus.edu.sg*
[§]*issty@mail.sysu.edu.cn*
[¶]*sysung@southtexascollege.edu*
[‖]*hxiong@andromeda.rutgers.edu*
[**]*chaost@dgut.edu.cn*

The duality between document and word clustering naturally leads to the consideration of storing the document dataset in a bipartite. With documents and words modeled as vertices on two sides respectively, partitioning such a graph yields a co-clustering of words and documents. The topic of each cluster can then be represented by the top words and documents that have highest within-cluster degrees. However, such claims may fail if top words and documents are selected simply because they are very general and frequent. In addition, for those words and documents across several topics, it may not be proper to assign them to a single cluster. In other words, to precisely capture the cluster topic, we need to identify those micro-sets of words/documents that are similar among themselves and as a whole, representative of their respective topics. Along this line, in this paper, we use hyperclique patterns, strongly affiliated words/documents, to define such micro-sets. We introduce a new bipartite formulation that incorporates both word hypercliques and document hypercliques as super vertices. By co-preserving hyperclique patterns during the clustering process, our experiments on real-world data sets show that better clustering results can be obtained in terms of various external clustering validation measures and the cluster topic can be more precisely identified. Also, the partitioned bipartite with co-preserved patterns naturally lends itself to different clustering-related functions in search engines. To that end, we illustrate such an application, returning clustered search results for keyword queries. We show that the topic of each cluster with respect to the current query can be identified more accurately with the words and documents from the patterns than with those top ones from the standard bipartite formulation.

Keywords: Hyperclique pattern; pattern preserving; bipartite partitioning; co-clustering; topic extraction.

1. Introduction

In text categorization, typically the data are arranged as a word-document co-occurrence matrix. Most clustering algorithms focus on one-way clustering, i.e., cluster one dimension of the table based on similarities along the second dimension. For example, documents may be clustered based upon their word distributions. However, the inherent sparseness in such a high dimension is a big challenge for many conventional clustering algorithms. On the other hand, words can be clustered first according to their co-occurrence in documents, which may in turn improve the document clustering results with the reduced set of features.[1]

Such a duality between document and word clustering naturally raises a question: if they can be co-clustered at the same time. Such a problem can be formulated in an information theoretic framework, where the goal is to minimize the mutual information loss after representing the original data with their cluster labels.[2] Alternatively, the bipartite is also a competitive candidate for describing such a duality. With documents and words modeled as vertices on two sides respectively, the bipartite only allows edges linking different kinds of vertices.[3] 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 refer to those with highest within-cluster degrees.

However, such claims may fail if the cluster is not pure enough or it includes words/documents across multiple topics. Some documents are top simply because they contain many general words with high degrees. Others may span several topics and it is improper to give them a hard classification. When it comes to words, it gets worse. Quite a few words come with multiple meanings, hence it is unreasonable to classify them to a single class. For instance, given a collection of documents with topics including business and health, it may not be appropriate to assign word "cell" to a single class. In fact, it can appear in documents of any topic, with meaning "cell phone" or "cancer cell".

To perform natural clustering and to precisely capture the cluster topic, first we need to identify those micro-sets of words/documents that are very similar among themselves and, as whole, representative of their corresponding topics. Meanwhile, we need to ensure that they would not be separated into different clusters during the clustering process. Second, as for those documents and words across several topics, they should be allowed to go to more than one cluster. At first glance, the two objectives above appear in conflict. In the bipartite formulation, however, they can be both satisfied by making some super vertices that contain a micro-set of words/documents. The words/documents in the set should be highly affiliated and as a whole, be able to specify a topic clearly. For those words and documents across multiple topics, they can appear in more than one super vertex in different clusters.

Along this line, in this paper, we exploit hyperclique patterns[4] to define such micro-sets. 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 measure.[5] 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 by Xiong *et al.*[6] 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.

Indeed, to preserve patterns, the clustering algorithm should have the following two properties. First, it can be set in a way such that clustering begins with starting points that are either original objects or patterns. Second, it must not break up these patterns during the clustering process. According to these two properties, this paper proposes a new bipartite formulation for co-preserving patterns, where word hypercliques and document hypercliques are represented by super vertices on two sides of the bipartite respectively. Our approach, CO-preserving PAtterns in bipartite Partitioning(COPAP), is compared with the standard bipartite formulation on real-world document data sets from different domains. The experimental results show that we can make improvement on clustering results in terms of various external measures and the topic can be identified more precisely.

Finally, due to the high affiliation within hyperclique patterns, the pattern preserving partitioned bipartite naturally lends itself to various applications in search engines. For instance, instead of a long ranked list for keyword queries, it is better to return clustered search results by topics. This can be done by showing only the words and documents from the patterns, which are more compact and representative of those topics. To this end, we demonstrate an application of the COPAP method for returning clustered search results. We show that the topic of each cluster with respect to the current query can be identified more accurately with the words and documents from the patterns than with those top ones from the standard bipartite formulation.

Overview. The rest of this paper is organized as follows. Section 2 describes background and related work. In Section 3, we introduce the details of the COPAP method. Section 4 describes an application of the COPAP method in search engines. Experimental results of co-clustering are reported in Section 5, together with a demonstration on returning clustered search results. Finally, in Section 6, we draw conclusions and discuss future work.

2. Background and Related Work

In this section, we describe related work and introduce some background information including document clustering, graph based document clustering, hyperclique patterns, and pattern preserving clustering.

2.1. *Document clustering*

Clustering has been extensively studied in machine learning, data mining, and statistics fields.[7] In general, clustering algorithms can be divided into 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 also be categorized into these two classes.

Hierarchical clustering approaches falls into two categories: 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 found UPGMA to be the best in this class for clustering text.[8] As for the partitional clustering, probably K-means is the most widely used method. As a modification, bisecting K-means can also be employed in hierarchical clustering of documents and produces competitive results.[9,8]

2.2. *Graph based document clustering*

Graph-theoretic techniques have also been considered for clustering.[10] 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.[11]

Document clustering is based upon their word distributions, while word clustering is determined by co-occurrence in documents. Such a duality can be naturally modeled using a bipartite, where documents and words are modeled as vertices on two sides respectively.[3] 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. In addition, bipartite graphs have also been used to model other relationships, such as (documents, concepts),[12] (authors, publications),[13] and sentences from two news articles for correlated summarization.[14] Finally, the extensions to bipartite, k-partite graphs, have also been employed to formulate and learn the relationships between entities of multiple[15] and the same types.[16]

2.3. *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 $I = \{i_1, i_2, \ldots, i_n\}$ be a set of distinct items. Each transaction T in database D is a subset of I. We call $X \subseteq I$ an itemset. The support of X, denoted by $supp(X)$,

Table 1. Example word patterns from WAP.

pattern	*supp* (%)	*hconf* (%)
zdnet, wire	2.37	94.87
buckingham, palace	1.15	64.29
hong, kong	1.54	64.86
related, earlier, story	16.79	60.79
sportscaste, marv, assault	1.09	50.00
wire, pr, stock, quote, finance	3.40	34.64

is the fraction of transactions containing X. If $supp(X)$ is no less than a user-specified threshold, 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 an itemset X_1 implies the presence of the other itemset X_2 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 Ref. 4 Formally, the h-confidence of an itemset $P = \{i_1, i_2, \ldots, i_m\}$ 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.

Now let us have a flavor of hyperclique patterns. Table 1 shows example word hypercliques from WAP, a document dataset of YAHOO webpages under categories like business, online, people, etc. ZDNet is a website where business meets technology, so word "wire" appears frequently. The next two rows show two proper nouns. 'related earlier story' appears most often in news webpages. "sportscaste marv assault" reveals the event of former NBC sportscaster Marv Albert's sexual assault trial. The last row appears often on any financial news pages. 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 starting points, however, prevents them from being separated.

Next, we show why the hyperclique pattern of documents is a good pattern to preserve in clustering. Figure 1 illustrates the average entropy of the discovered document hypercliques from WAP for different minimum h-confidence and support

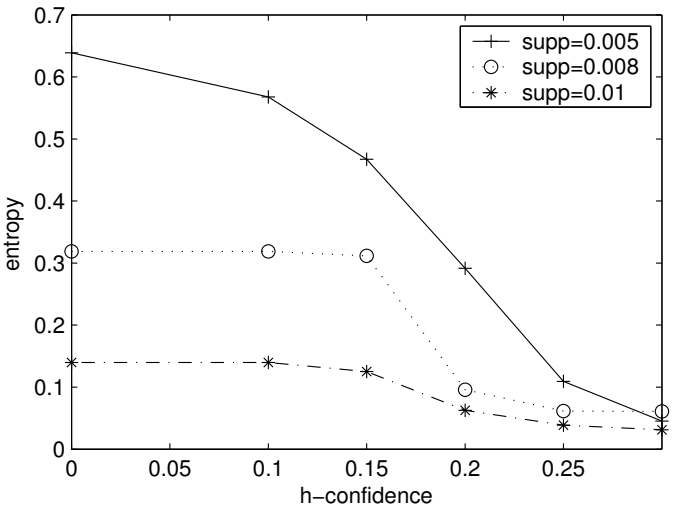

Fig. 1. Average entropy of document hypercliques from WAP.

thresholds. Note that when the minimum h-confidence threshold is zero, we actually have frequent itemset patterns instead of hyperclique patterns. We can see that as the minimum h-confidence threshold increases, the entropy of hyperclique patterns decreases dramatically, especially at low-support levels. This indicates that hyperclique patterns include objects from the same class above certain h-confidence levels. In contrast, the entropy of frequent patterns is high, especially for low support thresholds. This means that frequent patterns include objects from different classes. Thus, with respect to purity, the hyperclique pattern is a better candidate than frequent patterns for pattern preserving clustering.

2.4. *Pattern preserving clustering*

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, Xiong *et al.*[6] proposed the HIerarchical Clustering with PAttern Preservation (HICAP) algorithm and showed that its clusters are more interpretable than those of UPGMA. Compared to COPAP, 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 cost of hierarchical clustering practically prevents its use in large datasets, e.g., for search engines.

In addition, pattern preserving clustering is related to constrained clustering and frequent item set based clustering. Constrained clustering[17] 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.[18–20] 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.

2.5. *Applications 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 documents and leave it to the user to find which ones are of his/her 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. To circumvent this problem, Haveliwala[21] returns a set of topic sensitive lists by computing a set of PageRank vectors biased using a set of representative topics. Vivisimo[22] 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 even be meaningless. Here for each topic(cluster), we can 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.[23] 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.[24] 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.[25] With pattern preserving bipartites, we can make use of contextual information from both queries and retrieved documents.

3. COPAP: Co-Preserving Patterns in Bipartite Partitioning

Our approach COPAP is based on the bipartite graph partitioning with hyperclique patterns as super vertices. So the objects in the hyperclique pattern will not be separated during graph partitioning. Figure 2 gives the overview of the algorithm. Detailed description is given later in this section.

Input:
D: a data set represented by a word-document matrix.
α_w: a minimum support threshold for words.
θ_w: a minimum h-confidence threshold for words.
α_D: a minimum support threshold for documents.
θ_D: a minimum h-confidence threshold for documents.
K: the desired number of clusters.

Output: C: the resulting result.

Variables:
MD: the set of maximal document hypercliques.
LD: the set of documents not included in MD.
MW: the set of maximal word hypercliques.
LW: the set of words not included in MW.
BG: a bipartite graph.

Steps
1. MD= MaximalHyperclique Pattern(α_D, θ_D, D)
2. LD = UncoveredObjects(MD, D)
3. MW= MaximalHyperclique Pattern(α_w, θ_w, D^T)
4. LW = UncoveredObjects(MW, D)
5. BG = BipartiteGraph(MW, LW, MD, LD, D)
6. C = GraphPartition(BG, K)

Fig. 2. Overview of the COPAP algorithm.

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 significant words as features, which involves removing stopwords and those 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.

Given the word by document binary matrix A, if we treat words as transactions and documents as items, we can find maximal hyperclique patterns of documents. Next, we transpose A, where each row/transaction is for a document and each column/item for a word. In this case, we can identify maximal hyperclique patterns of words. For mining maximal hyperclique patterns, we employ a hybrid approach,[26] which exploited key advantages of both the depth first search strategy and the breadth first search strategy for efficient computation. 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.

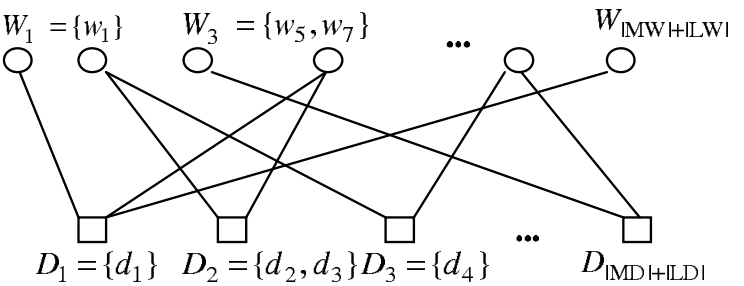

Fig. 3. The bipartite with meta-words and meta-documents.

3.2. *Generating the bipartite*

First some notations for general graph representation. A graph $G = (V, E)$ is composed of a vertex set $V = \{1, 2, \ldots, |V|\}$ 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 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 word hyperclique set MW and the document hyperclique set MD, we first identify those remaining words LW that never appear in MW and those remaining documents LD that never appear in MD. Then we construct vertex set $V = V_W \cup V_D$ as follows. V_W contains $|MW| + |LW|$ vertices each for a meta-word, i.e., either a word pattern in MW or a single word in LW. V_D contains $|MD| + |LD|$ vertices each for a meta-document, i.e., either a pattern in MD or a document in LD. An example bipartite is shown in Figure 3, where there are pattern vertices on both sides. The new $(|MW| + |LW|) \times (|MS| + |LS|)$ meta-word by meta-document matrix A' is defined in Eq. (1). That is, the association between meta-word W_i and meta-document D_j is the sum of association between all words w_k in W_i and all documents d_l in D_j.

$$A'_{ij} = \sum_{w_k \in W_i, d_l \in D_j} A_{kl} \,. \tag{1}$$

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 $V_1, V_2, \ldots, V_K$, by cutting the edges linking vertices in different parts. The general goal is to min-

imize 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 $min_{\{V_1, V_2, \ldots, V_K\}} \sum_{k=1}^{K} cut(V_k, V - V_k)$. 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

$$\sum_{k=1}^{K} \frac{cut(V_k, V - V_k)}{wgt(V_k)} . \tag{2}$$

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. For instance, the ratio cut,criterion[27] used for circuit partitioning, defines as $wgt(i) = 1$ for all vertices i and favors equal sized clusters. The normalized cut criterion,[28] used for image segmentation, defines as $wgt(i) = \sum_j M_{ij}$, which favors clusters with equal sums of vertex degrees, where vertex degree refers to the sum of weights of edges incident on it.

Finally, finding a globally optimal solution to such a graph partitioning problem is in general NP-complete,[29] though different approaches have been developed for good approximation solutions in practice.[30,31] Here we employ GRACLUS,[32] a fast kernel based multilevel algorithm, which involves coarsening, initial partitioning and refinement phases. Unlike other approaches such as METIS,[30] it 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. Applications to Clustered Search Results

The partitioned bipartite naturally lends itself to different clustering-related functions in search engines. In this section, we describe its applications to returning clustered search results. That is, given a keyword query, the search results are returned in a clustered format, where each cluster is represented by its topical documents and words.

As described in Figure 4, this job can also be done by the standard bipartite formulation. First we retrieve the set of documents $D(q)$ that contains query q and then partition it into groups $\{G\}$ according to the partitioned bipartite. The subsequent work is performed cluster by cluster. For representative documents, we directly select top documents from the cluster. When it comes to words, we give priority to those words shared by all documents in G(lines 5-8).

The counterpart in the bipartite with co-preserved patterns is more complicated, since we want to focus on those words/documents from patterns. The detailed

Input:
D: a dataset represented by a word-document matrix.
C: the partitioned bipartite containing the clustering results.
q: a query word.
k_1/k_2: the number of words/documents returned for each cluster.

Output: the clustered search results.

Steps
1. According to D, retrieve the documents $D(q)$ that contains q.
2. According to the cluster label in C, partition $D(q)$ into groups $\{G\}$, and keep those groups of size larger than k_2.
3. **For** each group G **Do**
4. Return top k_2 documents from G.
5. Compute $W(G)$, words shared by all documents in G.
6. **If** $|W(G)| \geq k_1$ **Then**
7. Return top k_1 words from $W(G)$.
Else
8. Return $W(G)$, and other top $k_1 - |W(G)|$ words from G.
End of for

Fig. 4. The standard bipartite algorithm for returning clustered search results.

procedure is shown in Figure 5. Within each group G/cluster, we first check if query q appears in (multiple)super vertex of word hypercliques. If yes, the words from the hypercliques receive priority of being selected(lines 5-6) and then we try to output any document that completely contains any single word hyperclique(lines 7-8). If not, we check if G contains(multiple)super vertex of document hypercliques. In this case, the document from the hypercliques are returned first(lines 11-12) and the words shared by such documents also get selected(lines 13-14). When the flow comes to line 16, it means that q appears in no word hypercliques and G contains no document hypercliques, then the word/document selection procedure is like the standard bipartite.

5. Experimental Evaluation

In this section, we present an experimental evaluation of COPAP. First we introduce the experimental datasets and cluster evaluation criteria, then we evaluate the clustering performance of COPAP against the standard bipartite formulation. Finally we illustrate clustered search results for some queries using the partitioned bipartite with co-preserved patterns.

Input:
D: a dataset represented by a word-document matrix.
C: the partitioned bipartite containing the clustering results.
q: a query word.
k_1/k_2: the number of words/documents returned for each cluster.

Output: the clustered search results.

Steps
1. According to D, retrieve the documents $D(q)$ that contains q.
2. According to the cluster label in C, partition $D(q)$ into groups $\{G\}$, and keep those groups of size larger than k_2.
3. **For** each group G **Do**
4. **If** q appears in (multiple)super vertex of word hyperclique $\{WH_i\}$ **Then**
5. Retrieve $W(G)$, the set of all words in $\{WH_i\}$. % *return words*
If $|W(G)| \geq k_1$ **Then**
Return top k_1 words from $W(G)$.
Else
6. Return $W(G)$, and other top $k_1 - |W(G)|$ words from G.
7. Retrieve $D(G)$,the set of document that contains at least one WH_i.
If $|D(G)| \geq k_2$ **Then**
Return top k_2 documents from $D(G)$.
Else
8. Return $D(G)$, and other top $k_2 - |D(G)|$ documents from $G - D(G)$.
9. **CONTINUE;**
10. **If** G contains(multiple)super vertex of document hyperclique $\{DH_i\}$ **Then**
11. Retrieve $D(G)$, the set of all documents in $\{DH_i\}$. % *return documents*
If $|D(G)| \geq k_2$ **Then**
Return top k_2 documents from $D(G)$.
Else
12. Return $D(G)$, and other top $k_2 - |D(G)|$ documents from $G - D(G)$.
13. Compute $W(G)$, the set of word shared by $D(G)$.
If $|W(G)| \geq k_1$ **Then**
Return top k_1 words from $W(G)$.
Else
14. Return $W(G)$, and other top $k_1 - |W(G)|$ words from $D(G)$.
15. **CONTINUE;**
16. **CALL** fragment (lines 4-8) in the procedure for standard bipartite.
End of for

Fig. 5. The co-preserving bipartite algorithm for returning clustered search results.

5.1. *Experimental datasets*

In our experiments, we used six datasets. To ensure diversity in the datasets, we obtained them from different sources. The RE0 and RE1 data sets are from the Reuters-21578 text categorization test collection Distribution 1.0.[33] The data sets K1 and WAP are from the WebACE project;[34] each document corresponds to a web page listed in the subject hierarchy of Yahoo. WAP contains a finer-grain categorization than that in K1, where the entertainment category is divided into 15 classes further in WAP. Datasets TR31 and TR41 are derived from TREC-6 and TREC-7

Table 2. Characteristics of data sets.

data	RE0	RE1	K1	WAP	TR31	TR41
#doc	1504	1657	2340	1560	927	878
#word	2886	3758	4592	8460	4703	7454
#class	13	25	6	20	7	10
MinClass	11	13	60	5	2	9
MaxClass	608	371	1389	341	352	243
min/max	0.018	0.035	0.043	0.015	0.006	0.037

collections.[35] The classes of these datasets correspond to the documents that were judged relevant to particular queries. For all data sets, we used a stoplist to remove common words, stemmed the remaining words using Porter's suffix-stripping algorithm[36] and removed those words with extreme low document frequencies. Some characteristics of these data sets are shown in Table 2.

5.2. *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 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 defined as

$$NMI = \frac{H(T) + H(C) - H(T,C)}{\sqrt{H(T)H(C)}}, \tag{3}$$

$$CE = H(T|C) = H(T,C) - H(C) \tag{4}$$

where $H(X)$ denotes the entropy of X and $\log_2$ is used here in computing entropy. NMI measures the shared information between T and C and it reaches the maximal value of 1 when they are the same. CE tells the information remained in T after knowing C and it reaches the minimal value of 0 when they are identical. 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.[37] 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:

$$R_{ij} = C_{ij}/C_{+j}\,, \tag{5}$$

$$P_{ij} = C_{ij}/C_{i+} \tag{6}$$

where C_{+j}/C_{i+} is the sum of jth column/ith row, i.e., jth class size/ith 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 Eq. (7).

$$F_{ij} = \frac{2R_{ij}P_{ij}}{P_{ij} + R_{ij}}\,, \tag{7}$$

$$F = \frac{1}{n}\sum_{j} C_{+j} max_i\{F_{ij}\}\,. \tag{8}$$

As shown in Eq. (8), the overall value for the F-measure is a weighted average for each class, 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.

5.3. *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[26] 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 COPAP with the standard bipartite formulation on transactional data.

As for the graph partitioning criterion used in GRACLUS, 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 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 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 except for TR41, COPAP is able to achieve improvement on all datasets in terms of all four measures. The two parameters, support threshold and h-confidence threshold, were tuned separately for each dataset, but not

Table 3. Comparison on six datasets.

data	method	ERR	F	NMI	CE
RE0	COPAP	0.4109	0.3812	0.3288	2.385
	STD	0.4262	0.3341	0.2711	2.608
RE1	COPAP	0.4906	0.3983	0.3610	2.436
	STD	0.5214	0.3434	0.3434	2.800
K1	COPAP	0.1282	0.8734	0.6987	0.5676
	STD	0.1444	0.8097	0.6512	0.9000
WAP	COPAP	0.5147	0.4173	0.4615	0.9269
	STD	0.5551	0.3336	0.3677	1.125
TR31	COPAP	0.2808	0.5407	0.4411	1.600
	STD	0.3112	0.5177	0.3978	1.731
TR41	COPAP	0.2976	0.5129	0.4420	1.161
	STD	0.2654	0.6421	0.5657	1.499

Table 4. The confusion matrix and top words for K1.

	Health	Ent	Sports	Politics	Tech	Business
C_0	0	18	120	0	0	0
C_1	0	26	0	0	0	0
C_2	6	1210	21	46	0	2
C_3	2	5	0	63	0	4
C_4	0	130	0	5	60	136
C_5	486	1	0	0	0	0

C_0	game, championship, career, season, final
C_1	calendar, showbiz, anniversari, file, guarante
C_2	crash, princess, dodi, car, photograph
C_3	senat, republican, clinton, vote, democrat
C_4	stock, financi, analyst, internet, wire
C_5	risk, diseas, prevent, patient, medic

for each criterion. For instance, all results on TR31 in the table are recorded at $(supp = 0.06, hconf = 0.5)$ for word hypercliques and $(supp = 0.01, hconf = 0.7)$ for document hypercliques, though better ERR 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. Tables 4 and 5 show the confusion matrices for K1 and TR31. There are many zero entries and a number of rows have only one nonzero entry. Also shown are the corresponding top five words from each cluster, which clearly indicate the topics of some

Table 5. The confusion matrix and top words for TR31.

	301	306	307	304	302	305	310
C_0	7	87	0	5	2	0	0
C_1	27	0	74	7	10	0	1
C_2	92	11	0	19	6	15	0
C_3	3	4	13	79	7	3	0
C_4	3	97	1	1	0	0	0
C_5	105	0	1	4	1	0	0
C_6	115	28	22	36	37	3	1

C_0	rwanda, rwandan, un, pari, hutu
C_1	construct, gener, project, river, electr
C_2	length, docid, subject, column, edition
C_3	endang, speci, review, wildlife, anim
C_4	_an, attack, civilian, afr, african
C_5	spanish, colombia, jpr, td, colombian
C_6	growth, hard, econom, europ, market

clusters. Note that the original class labels from TR31 are numbers of particular queries. However, with top words, it is not hard to figure out certain queries. For instance, most documents of query 304 go to cluster C_3. Its topic must concern the reviews of the endangered wildlife, animals and species. Most documents of query 307 are assigned into cluster C_1 and it must talk about the construction project of electricity generator on a river.

5.4. *Applications to search engines: clustered search results*

In this subsection, we illustrate the application of the partitioned bipartite to showing clustered search results. The motivation is still the high affiliation within hypercliques.

We choose data K1 for this purpose, since its classes are easier to distinguish. First we manually select some words with multiple topics. Using them as queries, it is better to return clustered search results by topic, instead of a ranked flat list. Figure 6 shows some example query results of the co-preserving bipartite. The major differences from those of the standard bipartite will be highlighted later. For each cluster, we show the number of documents in that cluster, top five words, and the sentence where the query word appears in the top document. For clarity, the document class at a finer scale(20 classes in total) is also shown, e.g., E-online refers to class online under category entertainment.

As shown in Figure 4, the standard bipartite formulation can do this job by first grouping all the documents containing the query according to the cluster label, and then returning the top words and documents from each group of documents. In some cases, however, we find that its returned words are still too general, not closely enough related to the query. As for the bipartite with co-preserved patterns,

figure	blue
1. (27) percent inform online web billion E-online: according to **figures** by the Massachusetts-based research company 2. (6) clinton house washington bill elect Politics: who has **figured** large in questions about contributions to President Clinton's political campaigns 3. (16) compare women age adult posit Health: CDC officials credit these declining **figures** to early diagnosis and treatment 4. (78) award camera sound band televis E-industry: where the industry has struggled to **figure** out the hierarchy	1. (6) washington clinton house court diana E-people: The rest of her outfit from that night -- a cream colored camisole, white blouse, **blue** blazer and jeans -- was entered into evidence on Tuesday in the court room 2. (18) financi percent stock quote wire E-online: Jenkins said the company's research team succeeded in formulating a stable **blue** phosphor 3. (38) feature movie david comedi screen E-review: is complemented by tunes from Johnny Cash, Herb Alpert and the Tijuana Brass and **blue**s rocker Screamin' Jay Hawkins 4. (3) game season career victori left Sports: Brett Hull's third goal of the game with 2:44 left in overtime lifted the St. Louis **Blue**s to a 3-2 victory over the winless Los Angeles Kings
cell	digest
1. (6) finance stock analyst percent internet E-online: AT&T LAUNCHES POCKETNET INTERNET **CELL** PHONE 2. (127) normal gene brain professor cancer Health: The destruction and loss of key brain **cell**s that happen as a result of Alzheimer's disease may be reversible	1. (4) stock analyst chief chairman announc E-industry: Reader's **Digest** Reshuffles Executive Suite 2. (6) risk medic diseas find drug Helath: They also advocate the introduction of 'competitor' organisms into the **digest**ive tract of cattle
apple	model
1. (29) stock analyst finance comput cent E-online: The chairman of Dell Computer Corp. said Monday he would shut down **Apple** Computer Inc. if he were put in charge of it 2. (3) drug blood brain determine editori Health: including one in the fall of 1996 traced to unpasteurized **apple** juice 3. (7) award camera band broadcast art E-review: The film then shifts back to the present, though the teen Sweet William's body is still seen hanging lifelessly from the **apple** tree	1. (16) stock bureau chairman gate growth Business: WhoWhere? launched its generic free e-mail **model** through deals with Yahoo! rival Excite Inc. and other online communities in July and has more than a million users. 2. (10) protect respons risk diseas medicin Health: safety-oriented roadway design **model** which has proven successful in Europe 3. (7) celebr fashion gala crash paris E-people: Five-hundred **model**s appear in fashion show on the longest catwalk in the world

Fig. 6. Example clustered search results.

this problem is relieved considerably. For instance, given query "cell", the standard bipartite returned "risk, medic, diseas, find, drug" from the cluster of health. Obviously they are related to health and medicine, but not closely related to cell. The reason is that for the current group of documents containing word "cell", these words are still top, possessing the largest within-cluster degrees. In contrast, the bipartite with co-preserved patterns output "normal, gene, brain, professor, cancer", because each word forms a two-word hyperclique with 'cell', according to the steps (lines 5-6) in Figure 5 which dictate the words from the hypercliques receive priority of being selected. Words like "medicine" and "disease" are too general to be able to form a hyperclique with "cell", because $conf(\text{cell} \rightarrow \text{medic})$ is high, but not vice versa.

Similar observations were also made when there are no word patterns and we select top words from top/hyperclique documents. Given query "model", the standard bipartite only returned general words like "risk" and "disease" from the cluster of health. In contrast, the bipartite with co-preserved patterns output "protect, respons, risk, diseas, medicin", because the first two words come from a document hyperclique talking about road safety for drivers. Therefore, according to steps (lines 13-14) in Figure 5, they are selected first.

6. Concluding Remarks

In this paper, we presented a new approach, CO-preserving PAtterns in bipartite Partitioning(COPAP), for word-document co-clustering and cluster topic extraction. Hyperclique patterns capture strong connections between groups of objects and should not be separated during clustering. Using them as starting points in the bipartite, our experiments showed that better clustering results could be obtained in terms of various external criteria and the cluster topic can be identified accurately. Besides, the co-preserved patterns in the partitioned bipartite enable those words and documents across several topics to appear in more than one cluster as needed. Due to the unique structure of the partitioned bipartite, it naturally lends itself to clustering related functions in search engines. Finally we illustrated such an application, returning clustered search results for keyword queries. Experiments indicated that compared to the standard bipartite formulation, selecting topical words from word/document patterns is able to identify the topic that is more closely related to the current query.

In addition, the word patterns from partitioned bipartite graphs can also be used for topic-sensitive query expansion, such as search engine advertisement words. Often a few keywords become popular among advertisers and the bid price rises sharply. In such cases, it is important to find cheaper yet relevant keywords to bid on. Indeed, advertisers need to understand which magic words lead to more conversions with their potential customers, instead of with the general web users. The words can be infrequent in the users log, but must be highly affiliated with respect to the target group of customers. More generally, a good engine must enable adver-

tisers to target customers based on their search terms, pages visited, advertisements clicked and products purchased. Along this line, our future work will focus on incorporating demographic and behavioral targeting capabilities into the framework of existing search engines.

Acknowledgments

This work was partially supported by NSFC(No. 60673135 and 60373081), NCET, SRF for ROCS, Sci. Tech. Plan Foundation of Guangdong (No. 20070328005), and Sci. Tech. Plan Foundation of Dongguan (No. 2007108101022).

References

1. I. S. Dhillon, S. Mallela, and R. Kumar. A divisive information-theoretic feature clustering algorithm for text classification. *Journal of Machine Learning Research*, 3:1265–1287, 2003.
2. I. S. Dhillon, S. Mallela, and D. S. Modha. Information-theoretic co-clustering. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 89–98, 2003.
3. I. S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 269–274, 2001.
4. H. Xiong, P.-N. Tan, and V. Kumar. Mining strong affinity association patterns in data sets with skewed support distribution. In *Proceedings of the 3rd IEEE International Conference on Data Mining*, pages 387–394, 2003.
5. H. Xiong, P.-N. Tan, and V. Kumar. Hyperclique pattern discovery. *Data Mining and Knowledge Discovery Journal*, 13(2):219–242, 2006.
6. H. Xiong, M. Steinbach, P.-N. Tan, and V. Kumar. HICAP: Hierarchical clustering with pattern preservation. In *Proceedings of the 4th SIAM International Conference on Data Mining*, pages 279–290, 2004.
7. A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264 – 323, 1999.
8. Y. Zhao and G. Karypis. Evaluation of hierarchical clustering algorithms for document datasets. In *Proceedings of the 2002 ACM CIKM International Conference on Information and Knowledge Management*, pages 515–524, 2002.
9. M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. In *KDD Workshop on Text Mining*, 2000.
10. 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.
11. L. D. Baker and A. McCallum. Distributional clustering of words for text classification. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 96–103, 1998.
12. I. Yoo, X. Hu, and I.-Y. Song. Integration of semantic-based bipartite graph representation and mutual refinement strategy for biomedical literature clustering. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 791–796, 2006.
13. J. Sun, H. Qu, D. Chakrabarti, and C. Faloutsos. Relevance search and anomaly detection in bipartite graphs. *ACM SIGKDD Explorations*, 7(2):48–55, 2005.

14. Y. Zhang, C.-H. Chu, X. Ji, and H. Zha. Correlating summarization of multisource news with k-way graph bi-clustering. *ACM SIGKDD Explorations*, 6(2):34–42, 2004.
15. B. Long, X. Wu, Z. Zhang, and P. S. Yu. Unsupervised learning on k-partite graphs. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 317–326, 2006.
16. A. Strehl and J. Ghosh. Cluster ensembles - a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, 3:583–617, 2002.
17. A. K. H. Tung, R. T. Ng, L. V. S. Lakshmanan, and J. Han. Constraint-based clustering in large databases. In *Proceedings of the 8th International Conference on Database Theory*, pages 405–419, 2001.
18. F. Beil, M. Ester, and X. Xu. Frequent term-based text clustering. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 436–442, 2002.
19. J. Pei, X. Zhang, M. Cho, H. Wang, and P. Yu. Maple: A fast algorithm for maximal pattern-based clustering. In *Proceedings of the 3rd IEEE International Conference on Data Mining*, pages 259–266, 2003.
20. B. C. M. Fung, K. Wang, and M. Ester. Large hierarchical document clustering using frequent itemsets. In *Proceedings of the 3rd SIAM International Conference on Data Mining*, 2003.
21. T. H. Haveliwala. Topic-sensitive PageRank: A context-sensitive ranking algorithm for web search. *IEEE Transactions on Knowledge and Data Engineering*, 15(4):784–796, 2003.
22. Vivisimo. http://vivisimo.com/.
23. J. Xu and W. B. Croft. Query expansion using local and global document analysis. In *Proceedings of 19th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 4–11, 1996.
24. J.-R. Wen, J.-Y. Nie, and H.-J. Zhang. Clustering user queries of a search engine. In *Proceedings of the 10th International World Wide Web Conference*, pages 162–168, 2001.
25. 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. *Journal of the American Society for Information Science and Technology*, 54(7):638–649, 2003.
26. Y. Huang, H. Xiong, W. Wu, and Z. Zhang. A hybrid approach for mining maximal hyperclique patterns. In *Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence*, pages 354–361, 2004.
27. L. Hagen and A. Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE Transactions on CAD*, 11:1074–1085, 1992.
28. J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:888–905, 2000.
29. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Company, 1979.
30. G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998.
31. K. Andreev and H. Racke. Balanced graph partitioning. In *Proceedings of the Sixteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures*, pages 120–124, 2004.
32. I. S. Dhillon, Y. Guan, and B. Kulis. A fast kernel-based multilevel algorithm for graph clustering. In *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 629–634, 2005.

33. D. Lewis. Reuters-21578 text categorization text collection 1.0. http://www.research.att.com/∼lewis.
34. 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 *Proceedings of the 2nd International Conference on Autonomous Agents*, pages 408–415, 1998.
35. TREC. The text retrival conference. http://trec.nist.gov/.
36. M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.
37. R Baeza-Yates and B Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, 1999.