Mining Bipartite Graph using SCMiner

杨帆

June 19, 2013

Abstract

In this article the SCMiner, an algoritm aiming at extracting useful information from a large relational data set, is studied. The algorithm itself, including the original tests performed by the author is reimplemented and analyzed. There're also some extra test cases made and performed. Flaws and weakness of the original method were discovered during the research and experiments, and some effective improvements are proposed and proved to work. At the end of the article, more possible modifications are discussed and assessed.

1 Introduction

Many relations in real life are in nature bipartite. For example, the viewers opinions to movies, the choice of products by different customers, and the occurrence of different words in different articles. These data can be expressed naturally in bipartite graphs. However this faced mainly two problems. First, as the data are in large scale usually, it's unconvenient to display and visualize it. Second, the data is usually very noisy, while we hope we can extract the truly relevant information from it.

There're many related problems that we are interested in, for example, how many typical groups are there in these movie-goers? How can we recommend movies to a specific viewer? How to automatically discriminate articles related with different topics? If there's some mechanism to generate a summarization by clustering the original bipartite, that will be a great help. It may be used to study the main pattern of the data, and using the clustering information, we can recommend a user things that are interested by another user in the same cluster, for instance.

SCMiner, short for Summarization-Compression Miner, aims at solving these problems. The method was proposed by Feng et al and published on SIGKDD '12([1]). It compresses the original graph by simultaneousy clustering rows and columns of the data matrix. With the summarization graph it derives, insights into the data can be gained, and link prediction is easy and feasible.

2 SCMiner Algorithm

The Minimal Description Length(MDL) principle is a formalization of the Occam's Razor principle in which the best hypothesis for a given set of data is the one that leads to the best compression of the data [8]. It's an important criterion in data compression and data dimensional reduction. The introduce of it to bipartite mining was first seen in [5] in SIGMOD '08. In [1](the focus of this article), the idea is further developed. A novel method is proposed to simultaneously integrate data compression, link prediction and discovery of hidden structure.

SCMiner compresses a bipartite G into two sub-bipartite G_S and G_A, and a grouping information table. G_S stores only super nodes(clusters) and their connections, the grouping information table stores to which clusters all the normal nodes belong. G_A stores correction edges to record some details omitted in G_S, in order to recover G.

2.1 A quick example

In Figure 1, there's a bipartite showing people's interests in different topics. With the MDL criterion by SCMiner, a possible compression(and also the best one) of it should be like Figure 2.

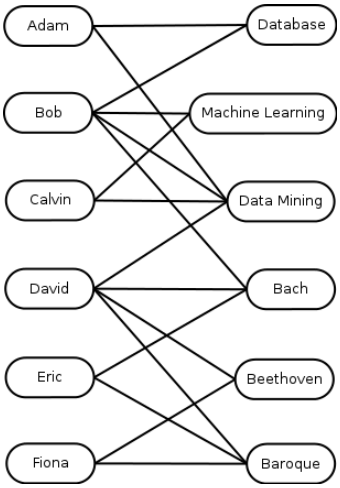


Figure 1: Original Bipartite

Figure 2 is the compressed representation. Figure 2a(the G_S) speaks the main pattern, while Figure 2b(the G_A) provides some correction. For example, the edge linking David to Data Mining means that, although in G_S David is a music lover but he still has relation with Data Mining. The edge linking Adam to Machine Learning suggests that though Adam likes CS, he doesn't do ML. Here in terms of SCMiner, Figure 2a is the Summary Graph, while Figure 2b is the Addition Graph. G_S, G_A together is a *summarization* of the original bipartite.

Intuitively, in Figure 1(The G), there're 16 edges. But in the compressed representation, there're only 8 edges in total. With the knowledge of G_S we are able to discover the hidden pattern in the mass data. By predicting missing edges, a recommendation system based on it is also possible(We will recommend Beethoven, instead of Database, to Eric). This example serves as a quick introduction to the idea of SCMiner, and in the following sections a formal definition about the algorithm will be presented.

2.2 MDL and Coding Cost

Assuming that a bipartite G = (V_1, V_2, E) is stored as an adjacency matrix A $\in |V_1| \times |V_2|$, the coding cost for a bipartite is calculated as follows, according to the lower bound of its entropy,

$$CC(G) = |V_1| \cdot |V_2| \cdot H(A) \tag{1}$$

where $H(A) = -(p_n(A) \cdot \log_2 p_n(A) + p_r(A) \cdot \log_2 p_r(A))$. $p_n(A)$ and $p_r(A)$ are the probabilities of finding 1 and 0 entries in A, or say, the probabilities of observing edges or not.

In the same manner, the coding cost for the grouping information table is as follows,

$$CC(ginfo) = \sum_{i=1}^{2} \sum_{j=1}^{N_i} |S_{ij}| \log_2 \frac{|V_i|}{|S_{ij}|} \tag{2}$$

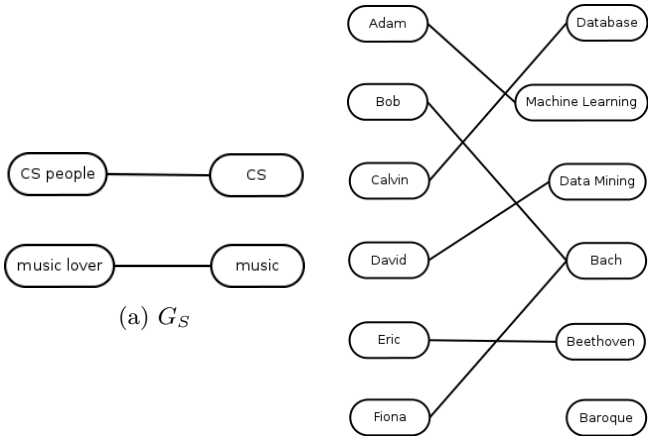
where N_i is the number of super nodes in type i, $|V_i|$ is the number of original nodes in type i and $|S_{ij}|$ is the number of members in super node S_{ij}.

Therefore the coding cost for the compressed representation of the previous example is

$$\begin{aligned} & CC(G_S) + CC(G_A) + CC(ginfo) \\ = \; & 2 \cdot 2 \cdot 1 + 6 \cdot 6 \cdot 0.65 + 6 \\ = \; & 33.4 \end{aligned}$$

2.3 The Algorithm

The SCMiner Algorithm can be divided into two aspects, the MDL principle and the heuristic method. The MDL principle is concerned with the minimal coding cost of the compressed representation, assuming that this is the best intepretion of the data, while the heuristic method is about searching and approaching it.



(a) G_S

(b) G_A

Node	Group	Node	Group
Adam	CS people	Database	CS
Bob	CS people	Machine Learning	CS
Calvin	CS people	Data Mining	CS
David	music lover	Bach	Music
Eric	music lover	Baroque	Music
Fiona	music lover	Beethoven	Music

(c) Groupinfo table

Figure 2: The compressed representation

The design of the heuristic method is very important, as a poor design may make the algorithm easily stuck in local optima. The basic idea is to merge nodes with similarity above a threshold th. If no nodes with that similarity are found, decrease th by a constant ϵ. The MDL principle is to choose a model with lowest cost. Repeat this process until th reach 0.

Hop2sim(hop 2 similarity) is used here to represent the similarity among nodes, defined as follows,

$$sim(S_{1i}, S_{1j}) = \frac{\sum_{k=1}^{n} |S_{2k}|}{\sum_{k=1}^{n+m} |S_{2k}|} \tag{3}$$

where S_{1i} means the i-th node in type 1 nodes, $S_{2k}(1 \leq k \leq n)$ are the common neighbors of S_{1i} and S_{1j}, $S_{2k}(1 \leq k \leq n+m)$ are all neighbors, and $|S|$ stands for the number of normal nodes contained in the super node S.

Normal nodes contained in a super node should have identical links. Therefore, before merging a group of nodes, their edges must be modified and unified. This, the *ModifyEdge* procedure, will be described later in the section.

The pseudocode of the SCMiner algorithm is provided in Algorithm 1.

The algorithm starts by searching and merging nodes with $th = 1$, i.e., completely identical nodes. Then the threshold th is gradually decreased. Groups of similar nodes found in the process are merged. The best summarization found is saved, and returned at the end.

In merging a group of nodes, there may be some neighbors that some of the nodes connect to, but some don't. To unify their linking pattern, for every such neighbor, either every node connect to it, or disconnect from it. The action of adding or removing edges is determine by the following criterion in Eq.5. This can be regarded as an approximation, instead of an actual calculation, of the possible increase in the coding cost of G_A. The action with lower cost is then chosen.

Algorithm 1 SCMiner

Input: Bipartite Graph G=(V,E), Reduce step ϵ
Output: Summary Graph G_S, Addition Graph G_A

//Initialization
$G_S = G, G_A = (V, \emptyset)$
Compute *mincc* using Eq.1 and Eq.2
$bestG_S = G_S, bestG_A = G_A$
Compute *hop2sim* for each S $\in G_S$ using Eq.3

//Searching for best Summarization
while $th > 0$ **do**
  for each node $S \in G_S$ **do**
    Get SN with $S' \in G_S$ and $hop2sim(S, S') < th$
  end for
  Combine SN and get non-overlapped groups *allgroup*
  for each $group \in allgroup$ **do**
    $\boldsymbol{ModifyEdge}(group, G_S, G_A)$
    Merge nodes of G_S with same link pattern
    Compute *cc* using Eq.1 and Eq.2
    Record $bestG_S, bestG_A$ and *mincc* if $cc < mincc$
  end for
  if $allgroup == \emptyset$ **then**
    $th = th - \epsilon$
  else
    $th = 1$
  end if
end while
return $bestG_S, bestG_A$

$$Cost_{remove} = \sum_{i=1}^{p} = \begin{cases} |S_{1i}| \cdot |S_{2k}| & \text{if } S_{1i} \text{ links to } S_{2k} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

$$Cost_{add} = \sum_{i=1}^{p} = \begin{cases} 0 & \text{if } S_{1i} \text{ links to } S_{2k} \\ |S_{1i}| \cdot |S_{2k}| & \text{otherwise} \end{cases} \tag{5}$$

The pseudocode of the procedure unifying linking pattern is provided in Algorithm 2.

Algorithm 2 ModifyEdge

Input: Group nodes *group*, G_S, G_A
Output: $G_S, G_A, hop2sim$

$allN$ = Neighbor(*group*)
cn = CommonNeighbor(*group*)
for Each node $S \in allN$ and $S \notin cn$ **do**
  Using Eq.5 to add or remove edge
  Add or remove edges in G_S
  Add additional edges in G_A
end for
Update *hop2sim* for each $S \in allN$ and $S \notin cn$
return $G_S, G_A, hop2sim$

3 Experiments and Modifications

This section provides experiments to empirically evaluate the performance of SCMiner on both synthetic and real data. Three synthetic cases described in the original paper are imple-

mented here. In addition, an unbalanced grouping case(idea of [9]), a noisy case, and an extreme case(both unbalanced and noisy) are also analyzed.

After the synthetic cases, a real-world dataset is tested, the Newsgroups datasets, containing 20,000 news articles of 20 different topics.

Some ideas to improve the performance came up during the experiments, and were also implemented and analyzed. Details are provided in this section.

3.1 Three original synthetic cases

The three original cases are shown in Table 1.

Dataset	S	C
BP1	$\begin{pmatrix} 0.8 & 0.2 \\ 0.1 & 0.9 \end{pmatrix}$	$\begin{pmatrix} 100 & 100 \\ 100 & 100 \end{pmatrix}$
BP2	$\begin{pmatrix} 0.9 & 0.8 & 0.1 \\ 0.1 & 0.9 & 0.8 \\ 0.1 & 0.2 & 0.9 \end{pmatrix}$	$\begin{pmatrix} 100 & 100 & 100 \\ 100 & 100 & 100 \end{pmatrix}$
BP3	$\begin{pmatrix} 0.8 & 0.7 & 0.2 & 0.8 \\ 0.9 & 0.3 & 0.8 & 0.2 \\ 0.3 & 0.8 & 0.2 & 0.7 \\ 0.9 & 0.8 & 0.7 & 0.2 \end{pmatrix}$	$\begin{pmatrix} 100 & 100 & 100 & 100 \\ 100 & 100 & 100 & 100 \end{pmatrix}$

Table 1: Three Original Cases

In matrix S, columns and rows represent the two types of clusters in the bipartite. S_{pq} denotes the percentage of links generated between clusters p and q. Entries in matrix C denote the size(number of normal nodes contained) of the two types of clusters.

A bipartite is then generated according to matrix S and C. A good clustering algorithm is assumed to correctly cluster the nodes and reveal the relation between these clusters. In the original paper, Normalized Mutual Information(NMI) [7] is used to evaluate the clustering quality. These synthetic testings are important, as they resemble real-world situations where some important inherent relations are heavily corrupted by noise. By implementing synthetic testings we will also have a more straightforward standard to evaluate the performance of the algorithm, and can better track its behaviour and do according improvements.

The result claimed by the author is provided in Table 2. Results of some other methods are also listed, like CA and ITCC, but they are to be discussed in later sections.

	BP1	BP2	BP3
SCMiner	1	1	0.9949
CA	0.6683	0.7897	0.8705
ITCC	1	1	0.8705
GS	0.2568	0.4069	0.5493

Table 2: Result stated in the paper

Here the results seem impressive. However, a reimplement of the experiment suggests that the performance is closely related with the choice of ϵ, the only parameter of SCMiner.

Take BP1 as an example, when ϵ is around 0.15, it yields perfect result, as is shown in Figure 3[1] However the clustering quality decreases when ϵ gets smaller. The clustering result is even worse when $\epsilon = 0.01$, shown in Figure 4.

In Figure 4, there're many small clusters that otherwise should be merged together. On the other hand, the MDL is 25767.47 in this case, while MDL = 24487.65 when $\epsilon = 0.15$. In fact,

[1]In these graphs, columns and rows stand for the two types of nodes in a bipartite. A colored rectangle shows the link between two super nodes. A linking graph preseves the original linking, while a grouping graph shows the connections in *bestGs* only.

(a) Linking Graph (b) Grouping Graph

Figure 3: Perfect result of BP1 with $\epsilon = 0.18$

(a) Linking Graph (b) Grouping Graph

Figure 4: Problematic result of BP1 with $\epsilon = 0.1$

it had been substantiated empirically in the paper that smaller MDL leads to better clustering quality. The relation between ϵ and MDL after many trials is shown in Table 3, which shows that SCMiner is not always yielding the best result with its heuristic searching and may fall in local optima.

ϵ	0.01	0.04	0.07	0.1	0.13	0.16
MDL	28200.49	25351.71	25110.22	24618.71	24618.71	24487.65(perfect)

Table 3: relation between ϵ and MDL

Same problems are encountered while testing BP2 and BP3. In fact, these local optima are resulted from a flaw in the heurisric searching algorithm by the original paper. In the next section, a modified heuristic method is proposed to remedy the problem.

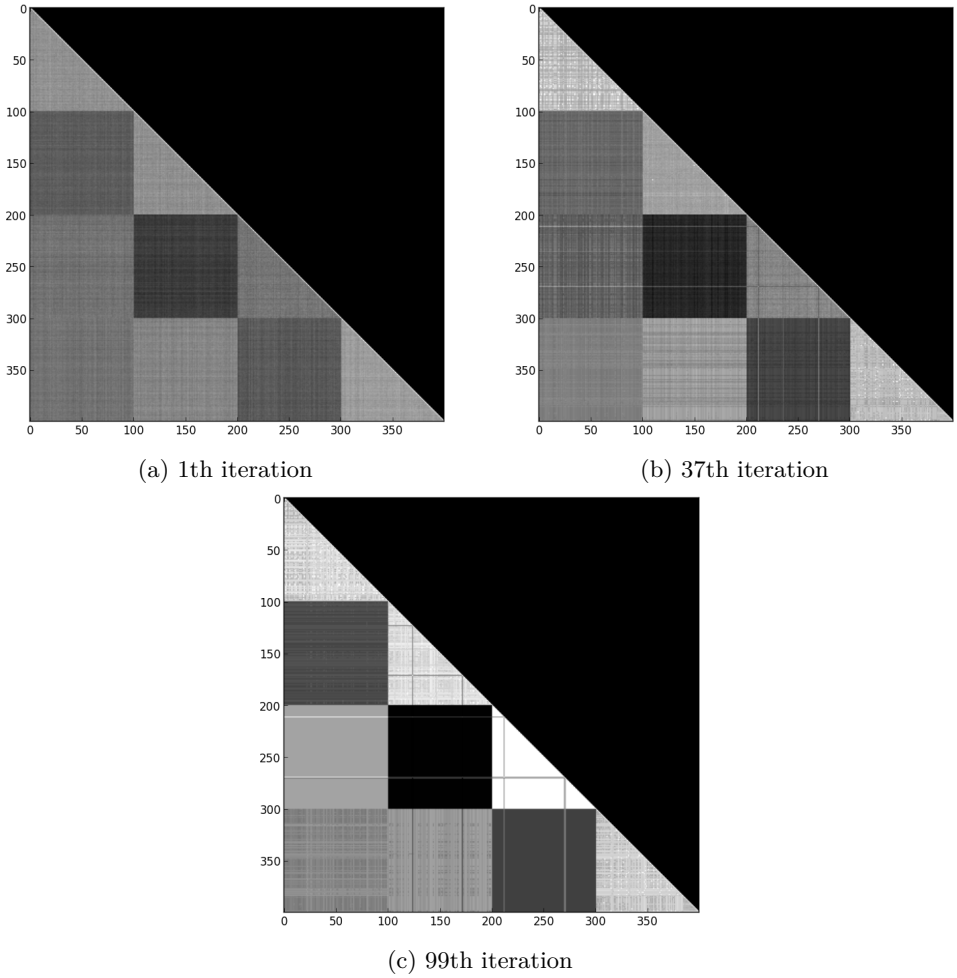
3.2 Modifications and Improvements

Figure 5 is the hop2sim matrix, showing the similarity among the type 1 node(i.e., the V_{1i}) in BP3. In BP3, nodes numbered from $100i$ to $100(i+1)(0 \leq i \leq 3)$ are in the same cluster, therefore in the hop2sim matrix, we should see area corresponding to these nodes lighter(a lighter dot means the corresponding two nodes have more indentical linking pattern).

In Figure 5b some unreasonable hop2sim values were derived(e.g., some white lines appears in the block corresponding to node 200∼300 and 100∼200, but they are not similar nodes), and were persisted to the end(these white lines still live in that block). It can be seen that in this specific case SCMiner failed to correct the errors it made previously. In fact, this phenomenon was repeatedly observed in many of the tests. According to the design by the original paper, when adding or removing edges, it pays cost according to the following:

$$Cost_{remove} = \sum_{i=1}^{p} = \begin{cases} |S_{1i}| \cdot |S_{2k}| & \text{if } S_{1i} \text{ links to } S_{2k} \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

$$Cost_{add} = \sum_{i=1}^{p} = \begin{cases} 0 & \text{if } S_{1i} \text{ links to } S_{2k} \\ |S_{1i}| \cdot |S_{2k}| & \text{otherwise} \end{cases} \tag{7}$$



(a) 1th iteration

(b) 37th iteration

(c) 99th iteration

Figure 5: Hop2sim matrix during a BP3 test

Therefore, it pays to make a mistake, but it also pays to correct the error. In this sense, I proposed an improved approach,

$$Cost_{remove} = \sum_{i=1}^{p} \sum_{V_{1j} \in S_{1i}} = \begin{cases} \begin{cases} -1 & \text{if } V_{1j} \text{ links to } S_{2k} \text{ in } G_A \\ 1 & \text{if } V_{1j} \text{ doesn't link to } S_{2k} \text{ in } G_A \end{cases} & \text{if } S_{1i} \text{ links to } S_{2k} \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

$$Cost_{add} = \sum_{i=1}^{p} \sum_{V_{1j} \in S_{1i}} = \begin{cases} 0 & \text{if } S_{1i} \text{ links to } S_{2k} \\ \begin{cases} -1 & \text{if } V_{1j} \text{ links to } S_{2k} \text{ in } G_A \\ 1 & \text{if } V_{1j} \text{ doesn't link to } S_{2k} \text{ in } G_A \end{cases} & \text{otherwise} \end{cases} \tag{9}$$

Intuitively, as the function of G_A is to explain details(in contrast to G_S, which is to express the main trend), if the modification(add or remove) calls G_S to add more explanation, then some cost should be paid. On the other hand, if the modification reduce the explanation introduced in G_A, it should be encouraged with a negative cost.

The modification has successfully solved previous problem, and another bonus it provides is that the algorithm now can produce more stable result with a much wider range of ϵ. Here

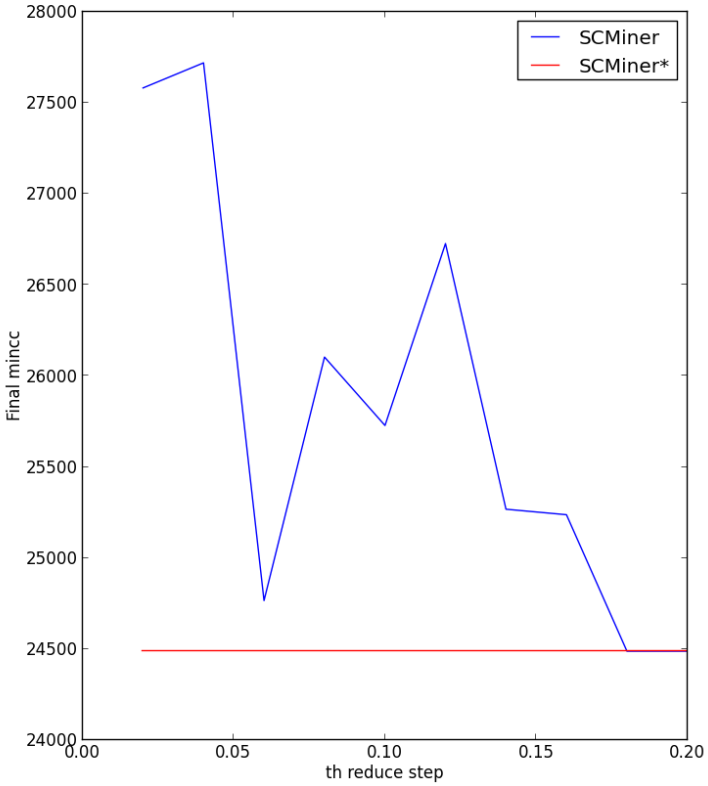


Figure 6: MDL by SCMiner and SCMiner*

in Figure 6 is a comparasion in BP1 with the original SCMiner, denoting the new version as SCMiner*.

Here in BP1, SCMiner* is always yielding perfect result and lowest cost no matter which ϵ is used, while SCMiner reaches the optimal with only some specific values of ϵ.

3.3 Three extra synthetic cases

These three cases, shown in Table 4, are harsher than the original ones. Some add addition noise and some introduce unbalanced structure. This challenges a lot to the original SCMiner. With $\epsilon = 0.1$ SCMiner* yeilds all perfect results, while SCMiner makes some mistakes and produces many small clusters.

3.4 Newsgroups dataset

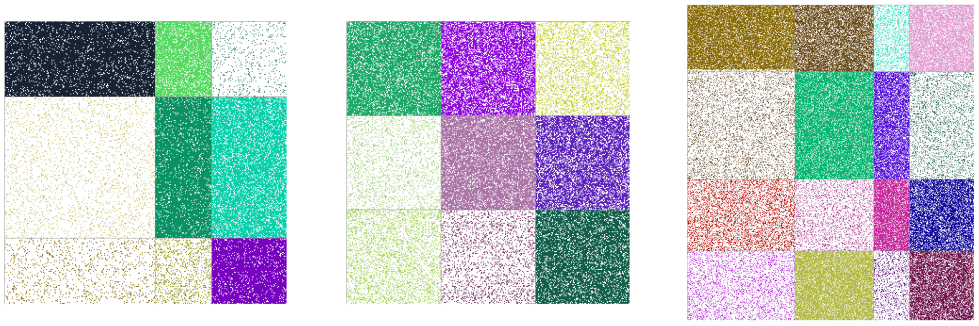
Though SCMiner(here SCMiner refers to the modified version, the SCMiner*) performs quite well on those synthetic datasets, we also want to know how SCMiner is doing on real data. Here the dataset Newsgroups, used in [9] for evaluating the SRE method, is tested, This dataset[2] contains 20,000 news articles, evenly divided into 20 different topics. The 20 group labels are listed as follows.

```
alt.atheism/
comp.graphics/
comp.os.ms-windows.misc/
comp.sys.ibm.pc.hardware/
comp.sys.mac.hardware/
comp.windows.x/
```

[2]Available at http://www.cs.cmu.edu/afs/cs/project/theo-11/www/naive-bayes.html

Dataset	S	C
BP4	$\begin{pmatrix} 0.9 & 0.8 & 0.1 \\ 0.1 & 0.9 & 0.8 \\ 0.1 & 0.2 & 0.9 \end{pmatrix}$	$\begin{pmatrix} 80 & 150 & 70 \\ 160 & 60 & 80 \end{pmatrix}$
BP5	$\begin{pmatrix} 0.8 & 0.7 & 0.3 \\ 0.2 & 0.8 & 0.7 \\ 0.3 & 0.2 & 0.7 \end{pmatrix}$	$\begin{pmatrix} 100 & 100 & 100 \\ 100 & 100 & 100 \end{pmatrix}$
BP6	$\begin{pmatrix} 0.8 & 0.7 & 0.3 & 0.7 \\ 0.2 & 0.8 & 0.7 & 0.2 \\ 0.3 & 0.2 & 0.8 & 0.7 \\ 0.2 & 0.8 & 0.2 & 0.7 \end{pmatrix}$	$\begin{pmatrix} 90 & 150 & 50 & 110 \\ 150 & 110 & 50 & 90 \end{pmatrix}$

Table 4: Three Extra Cases

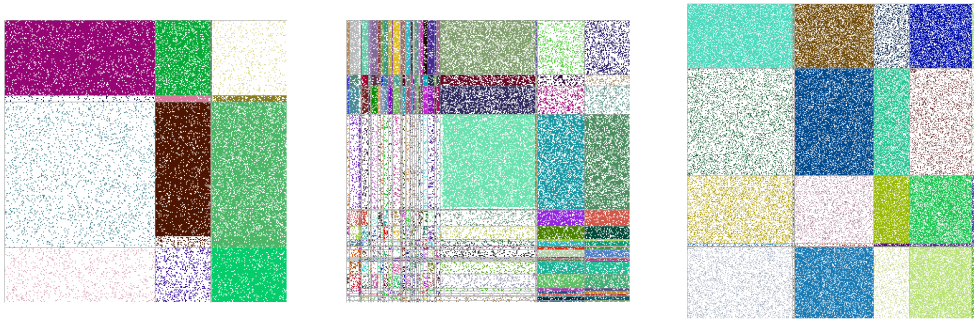


(a) BP4 (b) BP5 (c) BP6

Figure 7: Result produced by SCMiner*(Linking graph)

```
misc.forsale/
rec.autos/
rec.motorcycles/
rec.sport.baseball/
rec.sport.hockey/
sci.crypt/
sci.electronics/
sci.med/
sci.space/
soc.religion.christian/
talk.politics.guns/
talk.politics.mideast/
talk.politics.misc/
talk.religion.misc/
```

In fact, an real-world dataset used in the original paper is the MovieLens dataset. In that dataset, there're scores of movies(from 1 to 5) by many different users. However, it's not implemented here mainly because of two reasons. The first is that MovieLens dataset is a weighted bipartite instead of an unweighted one. Second, the records are not labeled, making them not suitable for clustering testing(they are suitable for evaluating recommendation system by guessing missing edges and weights). Also, it's claimed in the original paper that after deleting users voting for less than 30 movies and movies voted by less than 30 users, a 361×334 bipartite graph is obtained. However in actual testing, there're are far more nodes in the bipartite than that amount. The author didn't reply for my query in my email, and with the graph I generated by myself, the result is also not good, nearly nonsense(see the linking and grouping graph in Appendix).



(a) BP4 (b) BP5 (c) BP6

Figure 8: Result produced by Original SCMiner(Linking graph)

That's why the Newsgroups dataset is used. The preprocessing I did is as follows. First, stop words are striped from the articles. Then, all words are lemmatized(e.g, compiling -> compile, coded -> code). Finally, all words occurs less than 3 times in the whole corpus are deleted. The resulting bipartite is a article-word graph. An article node links to a word node if it's contained in the article.

Traditional weighting techniques for feature selection(like TF.IDF for instance) are not adopted here. In this sense, the unrelated terms(unimportant features) will be a challenge to SCMiner. It's worthwhile to mention that in [2], there're also no TF.IDF-like feature selections, and even stop words are counted. However as the method in [2] has considered the different information entropy contained in different terms, it still yeilds satisfactory result.

In the test, 100 articles from rec.sport.baseball/ and 100 from comp.graphics/ are mixed together(appearantly very different topics). SCMiner is expected to cluster them correctly into two groups. The ϵ is set to 0.1.

Result

The result is not satisfactory. Corresponding linking and grouping graph can be checked in the Appendix. In fact, there're numerous small groups(13 clusters with only size 1). A quick analysis shows that purity of these groups mostly range from 60% to 80%. However, as a comparasion, with the method proposed nin [9], the result is above 90%. The result and the comparasion has made it clearly that SCMiner failed to produce a satisfactory result on the Newsgroups dataset.

Analysis

Therefore it's necessary to analyze why SCMiner is not doing well. Reasons may be complex, but according to the design of the algorithm, which is divided into the MDL criterion, and the heuristic process, accordingly there may be two possible sources of problem. In another word, either the model did not heuristically reach an optimal enough MDL, or the MDL principle is in fact not applicable.

Figure 9 shows the variation of MDL during the test. The MDL is not reducing peacefully, and in contrast, there're even some jump-ups. For example the model experienced a sharp increase of MDL by 1500 bits, at position 652. It'll be helpful to understand this unusual behaviour, not aiming to fix this specific flaw, but to gain insights into the possible weakness of the algorithm.

In fact, at position 652, there're two article nodes merging. Here just call them node 122 and node 145, respectively. There're are 138 word nodes connected to article node 122, but 1563 word nodes connected to node 145. In these word nodes, the intersection of the 138 and 1563 words are mostly important key words(like img, graphic, package, display). Other words are either not related to the topic, or just rarely occur in the corpus(like gatekeeper, month, fractal).

Therefore, clearly, it's the unrelated and unimportant information that result in the increase of MDL, which can be one reason of the poor performance–SCMiner has to modify G_A greatly in order to account for the unimportant difference made by invaluable word nodes of the two article nodes. That's why SCMiner is doing well in graph with equally important nodes, but

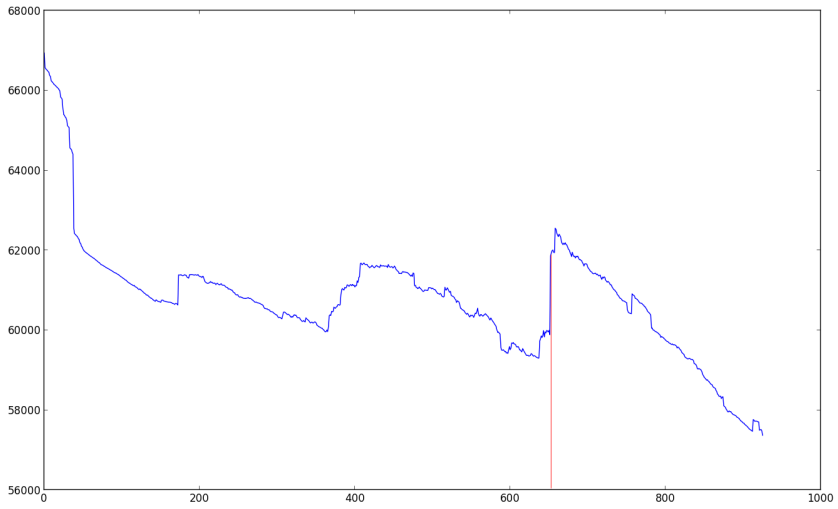


Figure 9: MDL during the test

performing poorly in real-world dataset like MovieLens and Newsgroups. There may be two possible solutions,

1. do TF.IDF like feature selection before clustering
2. modify the algoritm, making it suitable for nodes with various importance.

Method 2 is more favorable as method 1 introduces heavy parameterizing. In [2], different information entropy of nodes are taken into account, and that's why the method is performing well even with stop words.

Calculation of information entropy may be introduced by first deriving the information entropy of each edge, using

$$Entropy_{ij} = -(P \log_2 P + (1-P) \log_2 (1-P)) \tag{10}$$

$$P = \frac{P_r(i)|V1| + P_c(j)|V2|}{|V1| + |V2|} \tag{11}$$

Where $P_r(i)$ denotes the probability of observing ones in the i-th row of the data matrix, and $P_c(j)$ denotes the probability of observing ones in the j-th column.

This can be used when calculating hop2sim–If so, the heustic method is modified. The modified version had increased the purity by around 10%, but it suffers from two problems

- There're still small clusters with size 1
- The runtime therefore increased by nearly 3000% , making it unrealistic to perform lager-scale test.

As time was limited, no further optimization was tried then and details on implementation won't be shown here. More research on it is yet to do. And a test with TF.IDF feature selection should be also conducted to again prove that SCMiner *really* has no problem with balanced feature importances.

4 Complexity Analysis and Improvements

The most time-consuming step is the calculation of hop2sim(as can be seen in Figure 10). Assuming there're N vertices, the whole runtime complexity is $O(N \cdot d_{av}^3)$, where d_{av} denotes the average number of neighbors of a node. As only affected hop2sims are updated in each merge, the

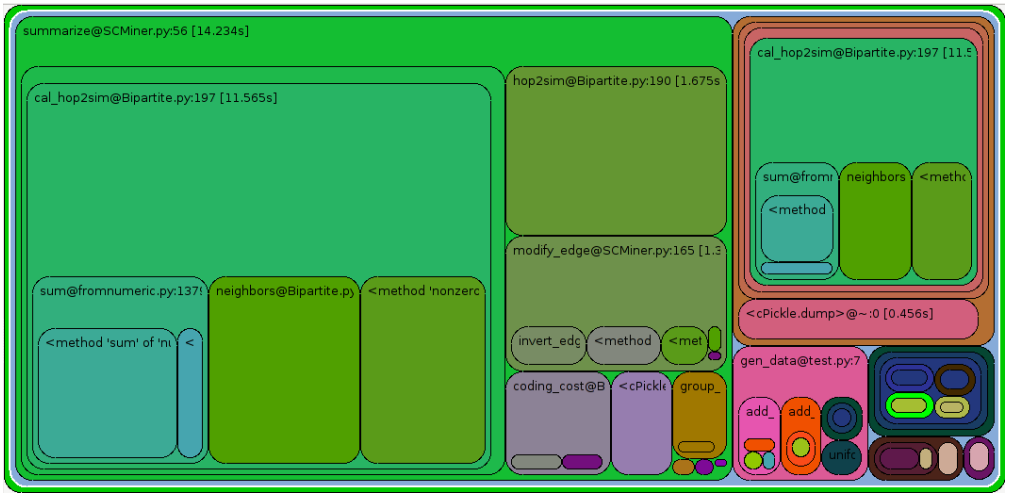


Figure 10: Runtime profile of different modules

actual complexity is $O(d_{av}^4)$ on average. There can be at most N merges, so the whole complexity is $O(N \cdot d_{av}^4)$.

In this sections, some tricks on optimizing the original algorithm are discussed. To skip these trival details, readers may jump to the next section in this article. Typically, with the unoptimized code, it takes as long as 60 seconds to do a BP2 test, while it costs only 7 seconds with optimized one.

Lazy hop2sim update

In the original algorithm, at the end of the *ModifyEdge* process, hop2sims affected are updated. As is mentioned above, calculating hop2sim is very time-consuming. In fact, in a single loop, there may be many different groups being merge. In this sense, it's possible that some same hop2sims are calculated repeatedly in different merges. Therefore, in fact, in the *ModifyEdge* method, those affected nodes can be added into a set called *NodesToUpdate*. And those nodes are updated once only at the end of the loop.

An implement of this optimization sees a 3/4 cut on runtime.

Skipping Grouping cost

This idea is proposed in [5] which first introduce MDL to bipartite mining, however does not work well, and what's more, it mess up the result. The idea suggests neglect the grouping information cost, and only take the coding cost of G_S and G_A into account.

First, this doesn't improve the runtime(as it doesn't affect the runtime complexity). Second, with this modification that somehow violates the MDL pronciple, the result is not good. In the synthetic tests, resulting graphs are always containing very small clusters(see Figure 11). This phenomenon is also mentioned in [1].

Low-level Binary Operation Optimization

The first version of the software is written in Python, mainly using Python's Set and related set operations like intersection and union. However, as the model can be express in binary arrays, most operations can be translated into low-level binary operations. With empirical tests performed on BP1 to BP6, this boosts the efficiency by around 1000% .

5 Related Works

Many algorithms for bipartite mining have been proposed in the past decade. Some focus mainly on biclustering(or *co-clustering*) problem, a problem about simultaneously clustering rows and columns of a data matrix. There're also other method concentrate on link prediction like collaborative filtering and its variants.

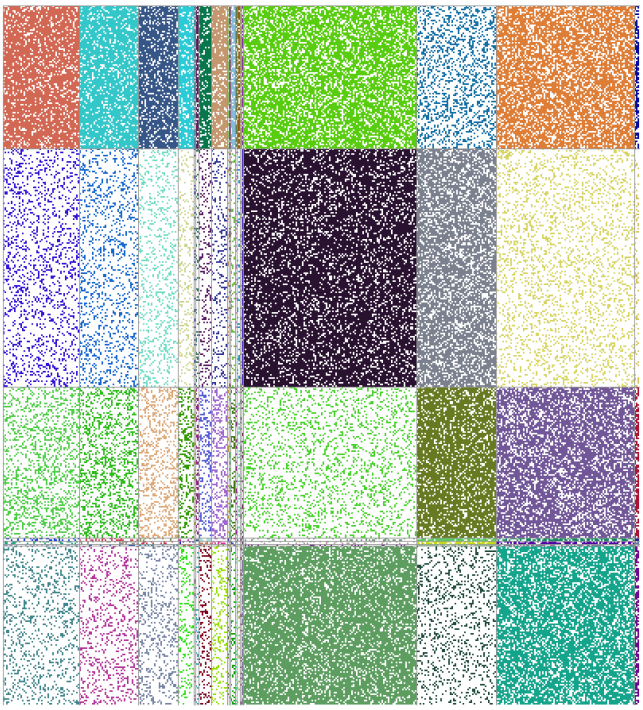


Figure 11: BP6 test without group info cost(Linking graph)

Co-clustering problem, in essence, is a dimensional reduction problem. Traditional matrix decomposition techniques like SVD and PCA(related with SVD) works very well on such problems, and the introduce of it into the co-clustering problem also sees satisfactory result([2] for example). This idea exits in many popular methods. There're also methods that integrate co-clustering and link prediction techniques. George, T. and Merugu, S.([4]) proposed a collaborative filtering framework based on SVD-related co-clustering. In [3], a method that introduce information theory into co-clustering is introduced by Dhillon.

Most clustering techniques produce *hard* partition, i.e., a node can belong to only one cluster. In [6], a co-clustering method based on bayesian network allows mixed membership of rows and columns.

The MDL method, as one of the traditional dimensional reduction techniques for data mining, is also not new. [9] first introduced it into bipartite mining. In that paper, the compression rate by the method is good, but unfortunately this doesn't promise good clustering quality, and what's worse, it doesn't capture the semantic structure of the bipartite.

6 Further Works and Improvements

As is stated before, nodes and edges shouldn't be treated equally. Some of them are more important. Therefore it will be helpful to introduce information entropy into the calculation. With some optimization, the result may further improve.

Besides MDL and SVD, people also use AutoEncoder to do dimensional reduction. AutoEncoder, one building block of deep neural network in terms of deep learning, aim at producing encoder to automatically compress data. They are to be trained with network training algorithm like backpropagation.

There's a bottleneck in the hidden layers(shown in Figure 12), resulting in information loss through the flow from the input to the output. This is deliberately. In the hidden layers, data are represented in a compressed form, and hopefully we will get the output that represents the most significant pattern of the data.

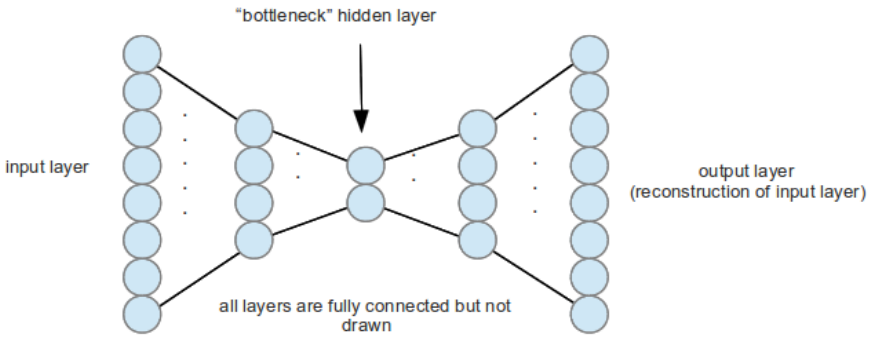
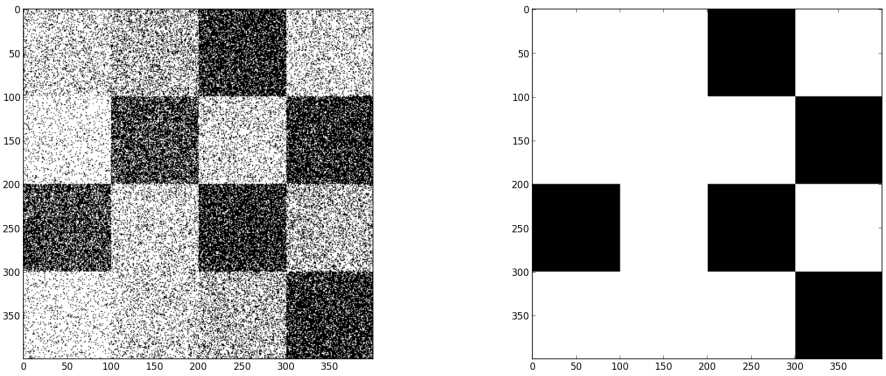


Figure 12: Structure of an AutoEncoder

We may introduce AutoEncoder into bipartite clustering in this way: input the data matrix into the network row by row, and train the model to best approximate the rows at the output. The network may in this way *remember* the most important pattern of the bipartite.

A quick implementation shows that AutoEncoder works well for the bipartite mining problem. For example, the following(Figure 13) is the input and output signals for BP3.



(a) Input Signals (b) Output Signals

Figure 13: A test of AutoEncoder for bipartite mining problem

7 Conclusion

In this research, the original SCMiner is studied and implemented. Some testing is performed to prove its effectiveness. During the research, some weakness of the original algoritm is discovered, and corresponding remedies are proposed and proved to work.

This research had exposed me to some common techniques used in especially in data dimensional reduction. Some essential knowledge like SVD and MDL was understanded and practiced during the process. Further works may include improving the orginal algorithm with more entropy calculation, and some possible integration of dimensional reduction techniques commonly used in the deep learning area.

References

[1] Jing Feng, Xiao He, Bettina Konte, Christian Böhm, and Claudia Plant. Summarization-based mining bipartite graphs. In *Proceedings of the 18th ACM SIGKDD international*

conference on Knowledge discovery and data mining, KDD '12, pages 1249–1257, New York, NY, USA, 2012. ACM.

[2] Inderjit S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '01, pages 269–274, New York, NY, USA, 2001. ACM.

[3] Inderjit S. Dhillon, Subramanyam Mallela, and Dharmendra S. Modha. Information-theoretic co-clustering. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '03, pages 89–98, New York, NY, USA, 2003. ACM.

[4] T. George and S. Merugu. A scalable collaborative filtering framework based on co-clustering. In *Data Mining, Fifth IEEE International Conference on*, pages 4 pp.–, 2005.

[5] Saket Navlakha, Rajeev Rastogi, and Nisheeth Shrivastava. Graph summarization with bounded error. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, SIGMOD '08, pages 419–432, New York, NY, USA, 2008. ACM.

[6] Hanhuai Shan and A. Banerjee. Bayesian co-clustering. In *Data Mining, 2008. ICDM '08. Eighth IEEE International Conference on*, pages 530–539, 2008.

[7] Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: is a correction for chance necessary? In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 1073–1080, New York, NY, USA, 2009. ACM.

[8] Wikipedia. Minimum description length — wikipedia, the free encyclopedia, 2013. [Online; accessed 7-June-2013].

[9] Hongyuan Zha Xiaofeng, Xiaofeng He, Chris Ding, Horst Simon, and Ming Gu. Bipartite graph partitioning and data clustering. In *In CIKM*, pages 25–32, 2001.

A The result on MovieLens data

Figure 14: The untrimmed whole data matrix(MovieLens)

In the resulting graph, similar rows and columns are grouped together.

B The result on Newsgroup data

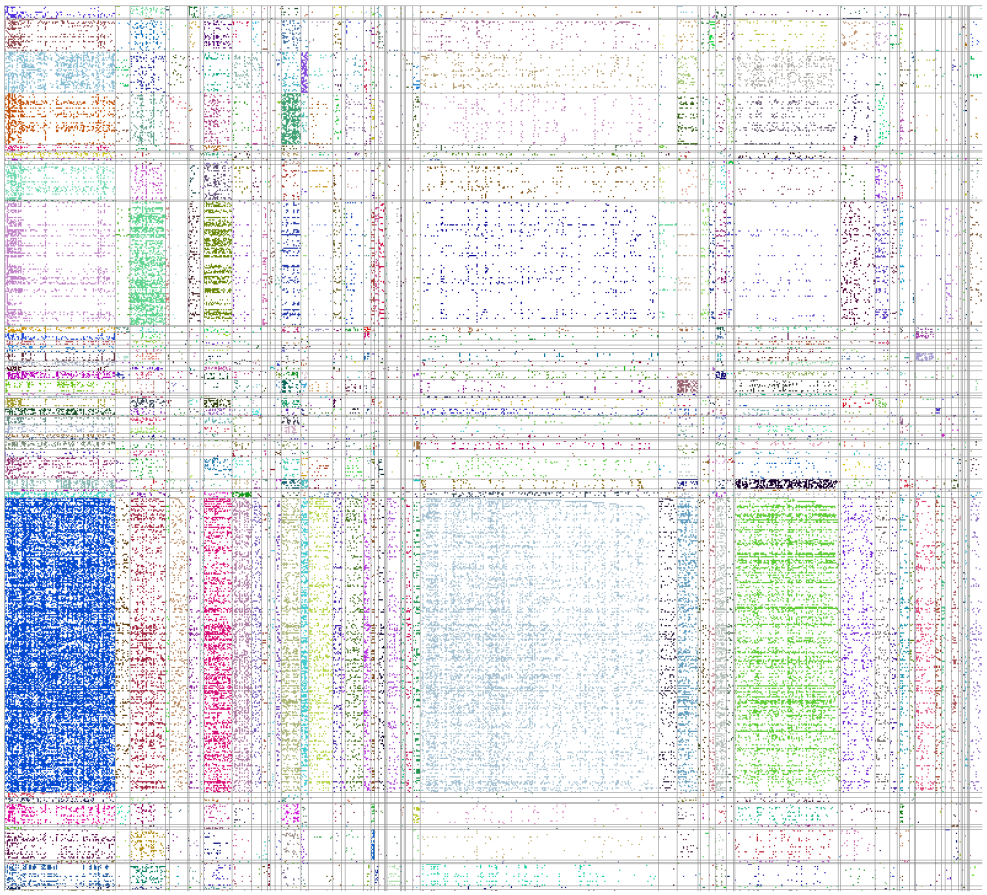


Figure 15: Linking graph of the result(MovieLens)

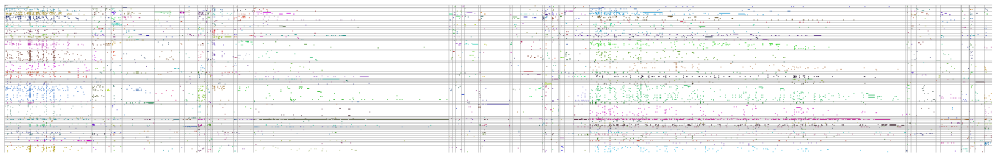


Figure 16: Linking graph of the result(Newsgroups)