Incremental Spectral Co-Clustering

Milind Prabhu
milind18@iitg.ac.in

Kushal Sangwan
sangwan18@iitg.ac.in

Amey Varhade
varhade@iitg.ac.in

ABSTRACT

Co-clustering is an important technique used not only for document clustering but also in customer relationship management applications, movie recommendation engines and in biomedical applications trying to classify patient symptoms and diagnoses. These applications see regular addition/deletion of data points and features to very large databases. and thus require these updates to be performed incrementally. Here we present one such algorithm for incremental co-clustering which is based on Inderjit S. Dhillon's algorithm [3]. It uses the static algorithm as a subroutine on the neighbourhood of newly inserted or deleted points , assuming that the change in labels occurs only till a particular depth from the inserted or deleted batch.It can also be proven that for a very large neighborhood , the clusters given by both static and incremental algorithms are exactly the same. This algorithm provides significant speedup as compared to the static version which is demonstrated in the experimental results.

1 INTRODUCTION

Clustering is the problem of clustering similar data points into one group and dissimilar ones into another. The problem of co-clustering is to simultaneously cluster data points as well as features.In document clustering problem, words that typically appear together are assumed to be associated with similar concepts, while documents having common words are supposed to have similar theme. For example, consider news article collection about different sports. If words batsman and bowler appear in an cricket article then they can be assumed to be similar .Same applies to the case when two football documents share the word goalie . Due to this duality between word and document clustering , co-clustering is widely used in for the problem of document clustering.

One of the first things to do for applying clustering algorithms to document datasets is to create vector space model [14]. The two steps involved in this are (1) extracting content bearing words from documents as features and (2) expressing each document as a vector of these features. Thus we can ultimately make a word-document matrix where the rows represent words and the columns represent document and the entries in the matrix represent the affinity(sometimes taken to be frequency itself) of a word W_i with a document D_j .

Spectral Co-Clustering algorithms generally do singular value decomposition on some normalized form of this word-document matrix which is followed by applying k-means algorithm on the obtained singular vectors. [3] [13] [10].All such algorithms work on a static snapshot of the document collection. But in real world scenario , these collections encounter frequent updates in form of insertion/deletion of documents as well as words. It is essential to perform these updates incrementally due to the large size of these datasets.However, there has been almost no work on incremental co-clustering algorithms which work on dynamic datasets.

Here we present one such incremental algorithm based on [3]. Dhillon's algorithm ,name it SCC, models the document-word matrix as a bipartite graph and then uses spectral techniques for partitioning it and thus obtaining it's clusters. The ISCC(incremental spectral co-clustering) algorithm presented her provides significant speed up with minimal accuracy loss as compared to the SCC algorithm.

The rest of this paper is organized as follows. In section 2, we briefly explain the SCC algorithm .We discuss related work on incremental co-clustering algorithms in section 3. The ISCC algorithm is presented in section4 with section 5 and 6 discussing the datasets used and the experimental results respectively. Section 6 concludes with lessons learnt from our project and section 6 discusses directions for future research.

Conference'17, July 2017, Washington, DC, USA
© 2021 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM...$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 RELATED WORK

2.1 Document Clustering in General

Document clustering is a well studied problem with a variety of formulations. The techniques used include agglomerative clustering methods [21], the heuristic k means method [17], latent semantic analysis [15] and the fuzzy c-means method [17]. There are many popular graph theoretic formulations of the document clustering problem. These are generally framed as optimization problems (that usually happen to be NP-hard) for which there are techniques to find approximate solutions. It turns out that these approximations turn out to be good enough in practice. Spectral techniques have widely been employed to find these approximate solutions and we describe these in the next subsection.

2.2 Work on Spectral Algorithms

Spectral clustering techniques were first proposed in 1973 [5] [6] when the connection between graph partitions and the eigen-vectors of the Laplacian was discovered. Over the next two decades these techniques were extended and improved. Spectral techniques became popular in the 2000s in machine learning circles due to the following works [16] [9] [8] [4]. We refer the interested reader to the tutorial by Luxbourg [20] on spectral clustering algorithms which includes a fairly detailed history of spectral clustering techniques and algorithms.

Our project is also based on an algorithm by Dhillon [3] which applies spectral techniques to the problem of Document-Word co-clustering. Algorithms before this paper considered partitioning problems on graphs where all the vertices represent documents and the weight of an edge between two documents to represent their similarity [1] [18]. A drawback of these methods was that the construction of these graphs required time quadratic in the number of documents. The algorithm we look at exploits the duality between word clustering and document clustering. A bipartite graph is constructed where the vertices are words and documents and all the edges are from documents to words with edge weights representing the association of the word with the document. This graph can be constructed asymptotically faster and a similar partitioning problem can be considered for it.

2.3 Incremental Algorithms

There are a wide variety of applications where there is a need for graph clustering algorithms where the graphs are dynamic and change slowly with time. A naive way to cluster these graphs is to recompute the clustering using one of the above static algorithms after fixed intervals of time. This can be very inefficient as most of the graph remains unchanged and therefore most of the computation performed is redundant. Therfore, incremental algorithms for spectral clustering have been proposed in the following papers [19] [11] [12] [7]. A common technique to “incrementalize" clustering algorithms that use spectral methods is to incrementally update the eigen-system using first order and second order derivatives of eigen values and eigen vectors as the graph is updated [11] [12].

Unfortunately, for incremental algorithms for spectral co-clustering have seen very little work. One of the related algorithms is LWI-SVD which works on updating the singular value decomposition for dynamic datasets[2]. It does low-rank approximations to speed up incremental SVD updates and uses multiple QR decompositions. As SVD is the bottleneck in spectral co-clustering, LWI-SVD combined with k-means can be used as incremental algorithm. [7].

3 THE SCC ALGORITHM

3.1 Mathematical Representation

The main idea of the algorithm is to exploit the duality between the document clustering and word clustering problem, i.e, one induces the other and thus better word clustering leads to better document clusters. The "best" for both cases will correspond to finding a partitioning of their bipartite graph with total weight of "cross" edges between partitions being minimum while maintaining the balance between partitions (i.e Normalized cut) for which the following algorithm is proposed.

Considering this, the intuition is to come up with a combined 'cost function' which captures both the aspects:

$$Q(V_1, V_2) = \frac{cut(V_1, V_2)}{weight(V_1)} + \frac{cut(V_1, V_2)}{weight(V_2)}$$

Using, the properties of the Laplacian **L**, the Weight matrix **W** and a generalized partition vector $\boldsymbol{q}$ we discover a equivalence between the Q function and a standard expression from linear algebra

$$\frac{\mathbf{q^\top L q}}{\mathbf{q^\top W q}} = Q(V_1, V_2)$$

The discreteness condition on q makes this optimization problem NP-hard. Finding the minimum for a real relaxation to expression on the left is what the following algorithm tries to do , but instead of using eigenvalues and eigenvectors, it tries to use the second left and right singular vectors of an appropriately scaled word-document matrix to yield good bipartitionings.

Example: See figure **??** for an example.

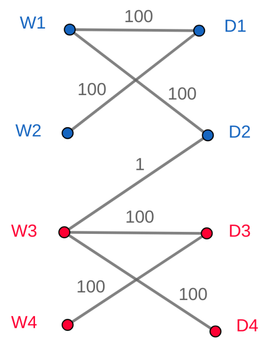


Figure 1: Example of optimal bipartitioning. All the vertices of a particular color represent a single cluster. The clustering shown in the figure minimizes the normalized cut value.

3.2 Algorithm Description

The **Bipartitioning algorithm** involves the following three steps.

(1) **Graph Representation:** Represent the data as a bipartite graph $G(U, V, E)$, the two disjoint sets U,V being the set of documents and the set of words appearing in these documents respectively. An edge between a document and word is assigned a weight which is a measure of the word's relation to the document (say, frequency of the word in the document). For convenience we let $|U| = w$ (the number of documents) and $|V| = d$ the number of words.

(2) **Singular Value Decomposition (SVD):** Let A be the the word by document matrix. Also, D_1 be a $w \times w$ diagonal matrix such that $D_1(i, i) = \sum_j A_{(i,j)}$ and D_2 be a $d \times d$ diagonal matrix D_2 such that $D_2(j, j) = \sum_i A_{(i,j)}$. The second step of the algorithm involves the computation of the second singular vectors u_2 and v_2 of the normalized matrix A_n where $A_n = D_1^{-1/2} A D_2^{-1/2}$. Here while u clusters the words , v clusters the documents.

(3) **Using k-means**: As the above vectors correspond to a real relaxation to the optimal generalized partition vector, one needs to find two bi-modal values to cluster using u and v. Run k-means on the one dimensional data $[D_1^{-1/2} u_2, D_2^{-1/2} v_2]$ to obtain these values and then finally the clustering.

The **Multipartitioning algorithm** is similar to the Bipartitioning algorithm. In step (2), $l = \lceil \log_2(k) \rceil$ singular vectors of A_n, $u_2, u_3 \cdots u_{l+1}$ and $v_2, v_3 \cdots v_{l+1}$ are computed. The l value corresponds to the fact in the optimal partitioning vector,we can have $2^l = k$ different values corresponding to the k different clusters. To obtain these values for the real relaxation solution obtained from step (2),in step (3) the k means algorithm is executed on

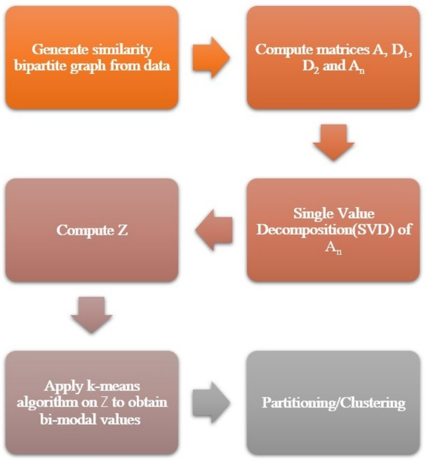

Generate similarity bipartite graph from data
Compute matrices A, D_1, D_2 and A_n
Single Value Decomposition(SVD) of A_n
Compute Z
Apply k-means algorithm on Z to obtain bi-modal values
Partitioning/Clustering

Figure 2: Static algorithm flowchart

the l-dimensional data $\begin{bmatrix} D^{-1/2}U \\ D^{-1/2}V \end{bmatrix}$ where $U = [u_2, u_3 \cdots u_{l+1}]$ and $V = [v_2, v_3 \cdots v_{l+1}]$.

4 INCREMENTAL SCC

4.1 Algorithm Description

Algorithm 1: Incremental Spectral Co-Clustering Algorithm

```
Input: The original word by document matrix $X$, sequence
  of updates (insertion/deletion of documents and words) in
  batches $B_1, B_2 \cdots B_m$
Output:Updated Clustering after each batch update $C^1$,$C^2$, .
  . . $C^m$
Run the spectral co-clustering algorithm on $X$ to get the
  initial clustering $C^0$
Get Bipartite word-document graph G from X
For $i = 1$ to $m$:
/* Here $B$ is a set of vertices(docs or words)
   that will be added or deleted                      */
    • Add $B$ to the graph $G$.
    • $G'$=get_neighbourhood($G$,$B$) /* $G'$ is obtained by
      performing a multi-source BFS from all vertices
      in $B$ upto depth d.                               */
    • Calculate connected components ($G^1, G^2, G^3 \cdots G^l$) of the
      subgraph $G'$.
    • Let X[1 2 . . . l] be an array of word document matrices
    • $C^i$= remove_neighbourhood($C^{i-1}$) /* Remove vertices
      in the neighbourhood from initial clustering, as
      their labels will be reassigned                    */
    • For $j = 1$ to $l$:
          $X[j]$=compute_word_document_matrix($G^j$).
          $C[j]$=spectral_co_clustering($X[j]$,k)
          $C^i$=merge($C[j]$ ,$C^i$)
          /* Merge in a way that minimizes the
      normalized cut of the graph.                       */
```

Intuition:The main intuition is that the insertion or deletion of node in a large graph, generally speaking, causes only local changes to the structure of clusters. Therefore, just checking a neighbourhood of the inserted or deleted vertex is enough to approximate the new cluster structure of the graph.

4.2 Insertion

Every Batch B contains the new vertices and their corresponding edges and these are added to the initial graph G. We get the d-neighbourhood (where depth d is a hyper parameter) of this batch using a multi-source BFS upto depth d from the vertices of batch B. This is in agreement with our assumption that insertion of vertices changes the cluster structures locally , and here this local region is spherical region of radius d from the batch B.

The vertices of this neighbourhood graph G' are then removed from the initial clustering C^{i-1} as the labels of these vertices are going to be reassigned.We separate this subgraph into connected components using a depth first search which is necessary for finding clusters as otherwise the SCC algorithm will just give the individual components as clusters.

Now these components are converted to word-document matrix form . Following this, the spectral co-clustering algorithm is used to partition every connected component by giving their word-document matrix as input .

The clusters found are then merged with the clustering C^i (points which are not in the neighbourhood) in such a way that normalized cut objective is minimized. For example if $k = 2$ and $C1$ and $C2$ be the initial clusters and $D1$,$D2$ be clusters obtained by partitioning one of the connected components of the neighbourhood graph , then one of $(D1 \cup C1, C2)$ and $(C1, D1 \cup D2)$ is chosen for which normalized cut value is less. Similar procedure is followed for D2 as well.

After all merges have been performed, the final clustering is obtained.

4.3 Deletion

Deletion is done in batches as well. The approach will be analogous to the one used for insertion. The neighbourhood of the vertices to be deleted is found by performing a multi-source BFS but the vertices themselves are then deleted from the neighbourhood.Rest of the implementation will be same as insertion case.

The vertices of this neighbourhood graph G' are then removed along with deleted vertices from the initial clustering C^{i-1}. The same procedure of separating the neighbourhood into connected components, partitioning them , and then merging them with the initial clustering is followed here as well.

4.4 Effect of Large Depth

Let us suppose that d_{max} be the depth when all vertices of the graph become part of the neighbourhood.Clearly d_{max} will be less then or equal to the diameter of the graph.

Now the clustering C^i obtained by removing the neighbourhood graph will become empty and this graph will have only one connected component i.e the full graph itself. Running the SCC algorithm on this graph here will yield cluster results identical to the static run of SCC but it is at the cost of speedup loss. The execution time here would be even more than the static run , as time is spent both in finding the neighbourhood and performing SCC on the full graph.

4.5 WorkFlow

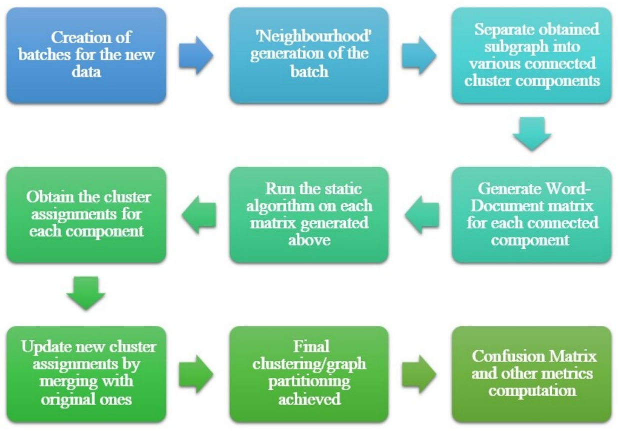
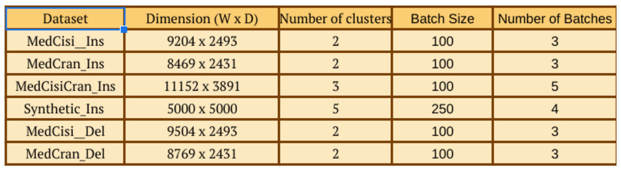

Creation of batches for the new data
'Neighbourhood' generation of the batch
Separate obtained subgraph into various connected cluster components
Generate Word-Document matrix for each connected component
Run the static algorithm on each matrix generated above
Obtain the cluster assignments for each component
Update new cluster assignments by merging with original ones
Final clustering/graph partitioning achieved
Confusion Matrix and other metrics computation

Figure 3: Incremental algorithm flowchart

5 DATASETS FOR INCREMENTAL ALGORITHM



Dataset	Dimension (W x D)	Number of clusters	Batch Size	Number of Batches
MedCisi_Ins	9204 x 2493	2	100	3
MedCran_Ins	8469 x 2431	2	100	3
MedCisiCran_Ins	11152 x 3891	3	100	5
Synthetic_Ins	5000 x 5000	5	250	4
MedCisi_Del	9504 x 2493	2	100	3
MedCran_Del	8769 x 2431	2	100	3

Figure 4: Dataset information

Figure 4 shows the different parameters concerning the datasets that we used to test our data. The Medline, Cisi and Cranfield datasets were obtained online(source) and combined to generate bipartitioning and multipartitioning datasets. For the insertion only datasets,some subset of vertices were included in the initial dataset and then rest of the vertices and their edges were added in multiple batches. The deletion datasets were similarly created by starting with the entire dataset and then words of the bipartite graph in batches.

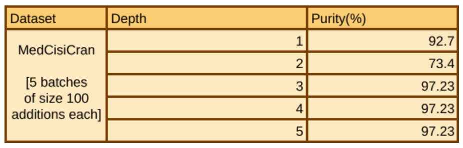
We created the synthetic datasets and these can be found on the github repository of the project (alongside other datasets that we used). The bipartite graphs were created by first dividing the 10000 words and documents into 5 clusters. Each edge within a cluster was added with probability $p_c = 0.1$ and edges between clusters were added with probability $p_n = 0.06$. The edges of the graph generated were then prepared accordingly.

6 EXPERIMENTAL RESULTS

The experiments were conducted on a machine with 8GB RAM and 2.20 GHz CPU. The 6 datasets used are given in Figure 4.

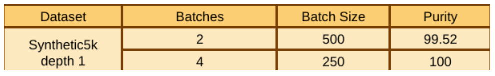
6.1 Insertion only Datasets

6.1.1 Depth Variation. The accuracy should increase with increasing depth but that was not observed .For example, in MedCisi_Ins(Figure 7) , the accuracy for depth 1 was 97.6% while for depth 2 was 93 %. One the reasons for this is that for odd depths we take some subset of documents out of the whole dataset and their corresponding words thus forming a sub-problem , but for even depths all words for the documents located at the maximum depth are not taken leading to their misclassification . Also larger neighbourhoods means larger dataset to run the SCC algorithm on, but this does not always guarantee an increase in accuracy.



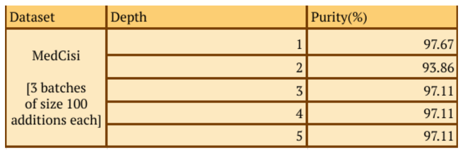
Dataset	Depth	Purity(%)
MedCisiCran [5 batches of size 100 additions each]	1	92.7
	2	73.4
	3	97.23
	4	97.23
	5	97.23

Figure 5: MedCisiCran Multipartitioning Results



Dataset	Batches	Batch Size	Purity
Synthetic5k depth 1	2	500	99.52
	4	250	100

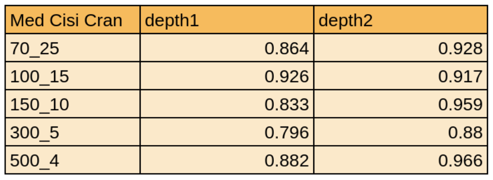
Figure 6: Synthtic5k dataset Multiipartitioning Results



Dataset	Depth	Purity(%)
MedCisi [3 batches of size 100 additions each]	1	97.67
	2	93.86
	3	97.11
	4	97.11
	5	97.11

Figure 7: MedCisi Bipartitioning Results

6.1.2 Batch Size Variation. Varying the batch sizes to include 70, 100,150, 300, 500 documents were taken for MedCisi_Ins, MedCran_Ins and MedCisiCraN datasets. When batch size is varied , if the ratio of increase in neighbourhood size is more than the increase in batch size then purity is going to increase otherwise its going to decrease. This leads to a pattern of increase and decrease being observed for purity values . For example, both in med_cran and med_cisi , going from batch size 300 to 500 leads to decrease in purity, cause 300 documents is close to 12% of total documents, and thus such large batch causes the neighbourhood size to saturate leading to its low increase when batch size is increased to 500.



Med Cisi Cran	depth1	depth2
70_25	0.864	0.928
100_15	0.926	0.917
150_10	0.833	0.959
300_5	0.796	0.88
500_4	0.882	0.966

Figure 8: MedCisiCran_Ins Results

6.2 Deletion only Datasets

Words here are deleted in batches of 100 . These deletions cause change in document labels. On an average one batch deletion causes the changing of 40 document labels. The observations here are taken from depth 2 as depth 1 will correspond to a neighbourhood of documents only , this is as the words themselves get deleted.

MedCran	70x5	100x3	150x2	300x1	500x1
Depth 4	99.38	99.3	99.38	99.13	98.43

Figure 9: MedCran_Ins Results

MedCisi	70x5	100x3	150x2	300x1	500x1
Depth 1	97.27	97.6	96.6	96.9	96.51

Figure 10: MedCisi_Ins Results

Dataset	Depth	Purity(%)
MedCisi (with Deletions)	2	95.78
	3	95.78
	4	96.35
	5	96.35

Figure 11: MedCisi dataset with deletion results

Dataset	Depth	Purity(%)
MedCran (with Deletions)	2	96.59
	3	97.3
	4	97.4
	5	97.4

Figure 12: MedCran dataset with deletion results

6.3 Comparison With Static Algorithm

As can be clearly seen from Fig.13, the execution time used of incremental algorithm is more than 60 percent less than that of the static algorithm with comparable accuracy results, specially for larger datasets. The memory consumed by the incremental algorithm is however 25 percent greater than used by the static one.

7 LESSONS LEARNT FROM THE PROJECT

- **Research:** Although a significant portion of our time was spent on reading papers and understanding existing algorithms, we got an opportunity to spend time thinking of new ideas. This gave us a flavour of what it is like to do research. We figured that coming up with new ideas is a way different (and way more exciting) thing that just verifying/learning existing ideas.
- **Working in teams:** Working in a team on such a project was very interesting. We learnt the importance of understanding the strengths of all the members of the team to optimize the total output of the team.
- **Organization:** There was a steep learning curve that had to be climbed to be more organized but it was worth it. Although there is still a long way to go, we now understand the importance of documenting progress consistently and using tools to organize our work.
- **Writing code:** We ended up writing about 2000 lines of code throughout the project both to prepare datasets and implement the algorithm. This might not be a huge codebase, but we realized the importance of writing readable and maintainable code since all three of us worked on writing the code. We also learnt to use the git version control system to effectively store/use different versions of the code.

8 FUTURE WORK

The ISCC algorithm can only handle insertions and deletions . The algorithm can be extended for updates in edge weights between words and documents. Also , insertions of documents and also their corresponding words does not affect the labels of points already present in the dataset but word only insertions can change the labels . It needs to be evaluated whether the algorithm works for these type of insertions.

Another interesting, but separate line of work would be to check if the algorithm can be extended to work on distributed systems. In such a setting there are multiple servers which receive the input data communicate information to a central server. The goal is to compute the clustering of all the data while minimizing the total communication to the central server.

Dataset	Incremental Purity(%)	Static Purity(%)	Static(mB)	Incremental(mB)	Static(ms)	Incremental(ms)
MedCisi	97.67	97.11	271	353	5123	1578
MedCran	99.57	99.38	273	344	4709	748
MedCisiCran	97.23	97.45	300	370	9468	3012
Synthetic5k	100	99.99	306	371	9234	2906

Figure 13: Comparison between Static and Incremental versions of algorithm

Link to the base paper	Link
Link to the code of the base paper	Link Our implementation
Link to the datasets used	Link This github repository contains all datasets used.
Link to your code	Link
Is your code working	Yes
What is the maximum speed up you are getting?	60%
What is the extra amount of memory required by your incremental algorithm?	25%
What is the accuracy of the incremental algorithm as compared to the static algorithm?	Short answer: Almost as good as the original algorithm. For more details please refer the above tables.
How did you measure the accuracy?	By calculating purity
Are you interested in further improving your work during the summer?	No. Would have been interested if not for other commitments.

Figure 14: Useful Information

REFERENCES

[1] Daniel Boley, Maria Gini, Robert Gross, Eui-Hong Sam Han, Kyle Hastings, George Karypis, Vipin Kumar, Bamshad Mobasher, and Jerome Moore. 1999. Document categorization and query generation on the world wide web using webace. *Artificial Intelligence Review* 13, 5 (1999), 365–391.

[2] Xilun Chen and K. Selcuk Candan. 2014. LWI-SVD: Low-Rank, Windowed, Incremental Singular Value Decompositions on Time-Evolving Data Sets. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '14)*. Association for Computing Machinery, New York, NY, USA, 987–996. https://doi.org/10.1145/2623330.2623671

[3] Inderjit S Dhillon. 2001. 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*. 269–274.

[4] Chris Ding and Xiaofeng He. 2004. Linearized cluster assignment via spectral ordering. In *Proceedings of the twenty-first international conference on Machine learning*. 30.

[5] WE Donath. 1973. Hoffman, a. *Algorithms for partitioning of graphs an, d computer logic based on eigenvectors of con, nection matrices* (1973), 938–944.

[6] Miroslav Fiedler. 1973. Algebraic connectivity of graphs. *Czechoslovak mathematical journal* 23, 2 (1973), 298–305.

[7] Tengteng Kong, Ye Tian, and Hong Shen. 2011. A fast incremental spectral clustering for large data sets. In *2011 12th International Conference on Parallel and Distributed Computing, Applications and Technologies*. IEEE, 1–5.

[8] Marina Meilua and Jianbo Shi. 2001. A random walks view of spectral segmentation. In *International Workshop on Artificial Intelligence and Statistics*. PMLR, 203–208.

[9] Andrew Ng, Michael Jordan, and Yair Weiss. 2001. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems* 14 (2001), 849–856.

[10] Feiping Nie, Xiaoqian Wang, Cheng Deng, and Heng Huang. 2017. Learning a structured optimal bipartite graph for co-clustering. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 4132–4141.

[11] Huazhong Ning, Wei Xu, Yun Chi, Yihong Gong, and Thomas Huang. 2007. Incremental spectral clustering with application to monitoring of evolving blog communities. In *Proceedings of the 2007 SIAM International Conference on Data Mining*. SIAM, 261–272.

[12] Huazhong Ning, Wei Xu, Yun Chi, Yihong Gong, and Thomas S Huang. 2010. Incremental spectral clustering by efficiently updating the eigen-system. *Pattern Recognition* 43, 1 (2010), 113–127.

[13] Manjeet Rege, Ming Dong, and Farshad Fotouhi. 2006. Co-clustering documents and words using bipartite isoperimetric graph partitioning. In *Sixth International Conference on Data Mining (ICDM'06)*. IEEE, 532–541.

[14] Gerard Salton and Michael J. McGill. 1986. *Introduction to Modern Information Retrieval.* McGraw-Hill, Inc., USA.

[15] Hinrich Schütze and Craig Silverstein. 1997. Projections for efficient document clustering. In *ACM SIGIR Forum*, Vol. 31. ACM New York, NY, USA, 74–81.

[16] Jianbo Shi and Jitendra Malik. 2000. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence* 22, 8 (2000), 888–905.

[17] Vivek Kumar Singh, Nisha Tiwari, and Shekhar Garg. 2011. Document clustering using k-means, heuristic k-means and fuzzy c-means. In *2011 International Conference on Computational Intelligence and Communication Networks*. IEEE, 297–301.

[18] Alexander Strehl, Joydeep Ghosh, and Raymond Mooney. 2000. Impact of similarity measures on web-page clustering. In *Workshop on artificial intelligence for web search (AAAI 2000)*, Vol. 58. 64.

[19] Christoffer Valgren, Tom Duckett, and Achim Lilienthal. 2007. Incremental spectral clustering and its application to topological mapping. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE, 4283–4288.

[20] Ulrike Von Luxburg. 2007. A tutorial on spectral clustering. *Statistics and computing* 17, 4 (2007), 395–416.

[21] Ellen M Voorhees. 1985. *The effectiveness and efficiency of agglomerative hierarchic clustering in document retrieval.* Technical Report. Cornell University.