CS568: Data Mining
Co-clustering documents and words using Bipartite Spectral Graph Partitioning[Review Report]

VARHADE AMEY ANANT(180101087), MILIND B PRABHU (180101091), and KUSHAL SANGWAN(180101096)

WHAT BOTTLENECK THIS CLUSTERING ALGORITHM ATTACKS?

This algorithm solves the co-clustering problem for words and documents together. The existing algorithms either cluster the documents based on word distributions or cluster words based on common occurrence in documents. This algorithm simultaneously clusters both words and documents through partitioning the corresponding bipartite graph. This is possible because of the duality between the word clustering and the document clustering i.e, related documents have common words and related words occur in the same document.

Also, existing graph theoretic algorithms for the document-clustering problem usually choose the vertices of the graph to represent documents and the weights on edges between vertices to be some measure of similarity between the documents. These algorithms take quadratic time in terms of the number of documents to construct the similarity graph which is computationally prohibitive. The algorithm proposed in the paper chooses the bipartite graph as the similarity graph between words and documents which can be constructed in linear time with respect to the number of documents.

WHAT IS THE OVERALL IDEA OF THE ALGORITHM? DO INCLUDE A FLOWCHART OR A FIGURE THAT EXPLAIN THE WHOLE ALGORITHM.

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

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

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.

The **Bipartitioning algorithm** involves the following three steps. Although the last two steps might seem very obscure they can be justified by some neat mathematics.

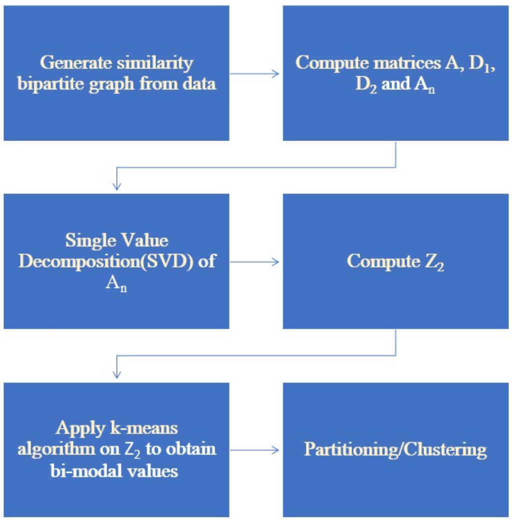
Generate similarity bipartite graph from data
Compute matrices A, D$_1$, D$_2$ and A$_n$
Single Value Decomposition(SVD) of A$_n$
Compute Z$_2$
Apply k-means algorithm on Z$_2$ to obtain bi-modal values
Partitioning/Clustering

Fig. 1. Overview of the bipartitioning algorithm

(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 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 $V = [v_2, v_3 \cdots v_{l+1}]$.

HOW IS THE EVALUATION OF THE ALGORITHM CARRIED OUT?

The algorithm is evaluated both for its bipartitioning and multipartitioning variants on large as well as small datasets. The paper shows the confusion matrices for the results of these tests and mentions that the goodness was gauged by the computation of the confusion matrix as well as by **purity** and **entropy** calculated from the confusion matrix.

Purity is an external evaluation criterion of cluster quality. It is the percentage of the total number of objects(data points) that were classified correctly. Entropy is another measure of performance of the algorithm. A higher entropy indicates that the cluster outputs by the algorithm in reality contains data points which belong to different labels or classes thereby saying that the algorithm is not doing well.

For the evaluation of the bipartitioning algorithm, the datasets Medline(medical abstracts), Cranfield(Aeronautical abstracts) and Cisi(information retrieval abstracts) were used. The tests were performed by mixing two of the above three datasets. In some cases care was taken to discard stop words and words that occurred either in a large fraction of the documents or in an insignificant number of documents. The authors however point out that the algorithm did well even when stop words were included.

For the multipartitioning algorithm, the evaluation was performed on a combination of the previous three datasets and also on a Yahoo datasets which consisted of news articles 6 categories: Business, Entertainment, Health, Politics, Sports and Technology. The documents in the Yahoo dataset were html pages and were pre-processed to discard HTML tags.

ARE YOU ABLE TO FIND OUT THE CODE AND DATASETS FOR THE EXPERIMENTS IN THE PAPER? IF NOT, HAVE YOU CONTACTED THE AUTHORS?

Primarily, two types of datasets are used, one for the bipartitioning and one for the multipartitioning varaint of the algorithm each. We were able to find some of the datasets used by the author:

- Bipartitoning Algorithm
  Cranfield, *Medline* and *Cisi*
- Multipartitoning Algorithm
  The above three datasets and the Yahoo dataset [Not Available].

The specific code used by the author was not available however we did find a *python implementation* of the algorithm in the sci-kit learn library.

COULD YOU FIND ANY REAL-WORLD APPLICATIONS OF THIS ALGORITHM? DOES ANY ML/DATA ANALYSIS PACKAGE INCLUDE THIS ALGORITHM?

The applications of the document clustering problem are:

- Biomedical applications to classify patient symptoms and medical diagnoses
- A customer relationship management (CRM) application where you want to co-cluster customers and items purchased.
- Movie recommendation engines engines co-cluster accumulated movie ratings from viewers. When a new viewer submits a score for a film she liked, the engine recommends other movies based on classifying the rating she provided to a cluster of audience movie ratings.
- Search engine optimization

The sci-kit learn python package has an implementation of the spectral co-clustering algorithm and it also cites this paper itself as the primary reference.

sci-kit learn 2.4.1 Special Co-Clustering