

# Spectral Methods for NLP and IR

EECS 767 – Advanced NLP

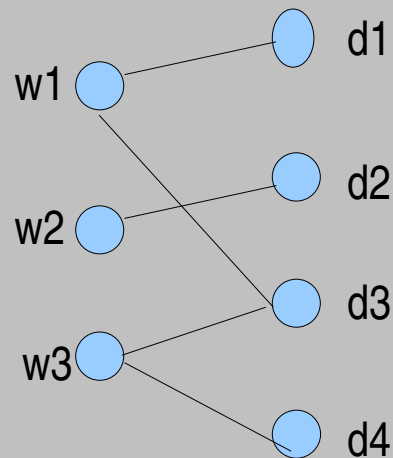
presented by Güneş Erkan

# Co-clustering documents and words using bipartite spectral graph partitioning

paper by Inderjit S. Dhillon

# Co-Clustering Documents and Words

- A collection of documents can be represented as a bipartite graph of words and documents.



m words:  $W = \{w_1, w_2, \dots, w_m\}$

n documents:  $D = \{d_1, d_2, \dots, d_n\}$

The corresponding m x n matrix:  $A$ .

The adjacency matrix of the bipartite graph:

$$M = \begin{bmatrix} \mathbf{0} & \mathbf{A} \\ \mathbf{A}^T & \mathbf{0} \end{bmatrix}$$

- Different choices for edge weights  $w_{ij}$ : tf, tf x idf, etc.

# Graph Partitioning and “Cuts”

- Suppose you partition the graph into sets (clusters) of nodes  $V_1, V_2, \dots, V_k$
- A good partitioning would have few inter-cluster links and many intra-cluster links.
- The “quality” of the partitioning can be measured in terms of the number of links between the clusters:

$$\text{cut}(\mathcal{V}_1, \mathcal{V}_2) = \sum_{i \in \mathcal{V}_1, j \in \mathcal{V}_2} M_{ij}.$$

$$\text{cut}(\mathcal{V}_1, \mathcal{V}_2, \dots, \mathcal{V}_k) = \sum_{i < j} \text{cut}(\mathcal{V}_i, \mathcal{V}_j)$$

# Duality of Word & Document Clustering

- Suppose you have a clustering of words  $W_1, W_2, \dots, W_k$ .
- Then you can use it to cluster the documents: put a document into the cluster that it has the most connections:

$$\mathcal{D}_m = \left\{ d_j : \sum_{i \in W_m} A_{ij} \geq \sum_{i \in W_l} A_{ij}, \forall l = 1, \dots, k \right\}$$

- And vice versa: If we had a document clustering, we could cluster the words.

# Duality of Word & Document Clustering

- “*Word clustering induces document clustering while document clustering induces word clustering*”.
- Does it sound familiar?
- Is there anything wrong with this assumption?
- Nice side effect: word clusters
- We want the minimum cut solution for this recursive

problem:

$$\text{cut}(\mathcal{W}_1 \cup \mathcal{D}_1, \dots, \mathcal{W}_k \cup \mathcal{D}_k) = \min_{\mathcal{V}_1, \dots, \mathcal{V}_k} \text{cut}(\mathcal{V}_1, \dots, \mathcal{V}_k)$$

# The Laplacian and Its Properties

- The Laplacian matrix:  $\mathbf{L} = \mathbf{D} - \mathbf{M}$ , where  $\mathbf{D}$  is the diagonal degree matrix, and  $\mathbf{M}$  is the adjacency matrix.
- Given a bipartitioning  $V_1, V_2$ , define the partitioning vector,  $\mathbf{p}$ : 
$$p_i = \begin{cases} +1, & i \in \mathcal{V}_1, \\ -1, & i \in \mathcal{V}_2. \end{cases}$$
- Nice property:  $\mathbf{p}^T \mathbf{L} \mathbf{p} = 4 \times \text{cut}(V_1, V_2)$
- $\mathbf{p}^T \mathbf{L} \mathbf{p}$  is zero when  $\mathbf{p}$  is all -1's or all 1's (putting everything into one cluster). So we need a better definition of a (minimum) cut.

# “Balanced” Cuts and Eigenvectors

- Suppose vertices have weights associated with them.
- We are looking for a cut that gives “balanced” clusters:

$$Q(\mathcal{V}_1, \mathcal{V}_2) = \frac{\text{cut}(\mathcal{V}_1, \mathcal{V}_2)}{\text{weight}(\mathcal{V}_1)} + \frac{\text{cut}(\mathcal{V}_1, \mathcal{V}_2)}{\text{weight}(\mathcal{V}_2)}$$

- When all weights are 1, minimize

$$\text{Ratio-cut}(\mathcal{V}_1, \mathcal{V}_2) = \frac{\text{cut}(\mathcal{V}_1, \mathcal{V}_2)}{|\mathcal{V}_1|} + \frac{\text{cut}(\mathcal{V}_1, \mathcal{V}_2)}{|\mathcal{V}_2|}$$

- When weight of a node equals its degree, maximize

$$S(\mathcal{V}_1, \mathcal{V}_2) = \frac{\text{within}(\mathcal{V}_1)}{\text{weight}(\mathcal{V}_1)} + \frac{\text{within}(\mathcal{V}_2)}{\text{weight}(\mathcal{V}_2)}$$

# “Balanced” Cuts and Eigenvectors

- Theorem 3: 
$$\frac{q^T L q}{q^T W q} = \frac{\text{cut}(\mathcal{V}_1, \mathcal{V}_2)}{\text{weight}(\mathcal{V}_1)} + \frac{\text{cut}(\mathcal{V}_1, \mathcal{V}_2)}{\text{weight}(\mathcal{V}_2)}$$

subject to  $q^T W e = 0$ , and  $q^T W q = \text{weight}(\mathcal{V})$

- for ratio-cut,  $W = D$ .

- Theorem 4: 
$$\min_{q \neq 0} \frac{q^T L q}{q^T W q}, \quad \text{subject to } q^T W e = 0$$

is solved when  $q$  is the eigenvector corresponding to the 2<sup>nd</sup> smallest eigenvalue of the eigenvalue problem

$$L z = \lambda W z \quad \text{or equivalently} \quad W^{-1} L z = \lambda z$$

# Connection to Singular Vectors

- When  $\mathbf{W} = \mathbf{D}$ , it turns out that  $\mathbf{W}^{-1} \mathbf{L} \mathbf{z} = \lambda \mathbf{z}$  can be reduced to two equations:

$$\begin{aligned} \mathbf{D}_1^{-1/2} \mathbf{A} \mathbf{D}_2^{-1/2} \mathbf{v} &= (1 - \lambda) \mathbf{u}, \\ \mathbf{D}_2^{-1/2} \mathbf{A}^T \mathbf{D}_1^{-1/2} \mathbf{u} &= (1 - \lambda) \mathbf{v}. \end{aligned}$$

- or equivalently  $\mathbf{A}_n \mathbf{v}_2 = \sigma_2 \mathbf{u}_2, \quad \mathbf{A}_n^T \mathbf{u}_2 = \sigma_2 \mathbf{v}_2$
- $\mathbf{u}$  and  $\mathbf{v}$  are singular vectors! remember HITS?
- Final solution:

$$\mathbf{z}_2 = \begin{bmatrix} \mathbf{D}_1^{-1/2} \mathbf{u}_2 \\ \mathbf{D}_2^{-1/2} \mathbf{v}_2 \end{bmatrix}$$

# Final Bipartitioning Algorithm

- Given a collection of documents, form the  $m \times n$  (*words x documents*) matrix  $\mathbf{A}$ .
- Compute  $\mathbf{A}_n = \mathbf{D}_1^{-1/2} \mathbf{A} \mathbf{D}_2^{-1/2}$
- Compute the singular vectors of  $\mathbf{A}_n$ :  $\mathbf{u}_2$  and  $\mathbf{v}_2$ , and form  $\mathbf{z}_2$
- Run the k-means algorithm on  $\mathbf{z}_2$

# Multipartitioning algorithm

- Run bipartitioning algorithm recursively.
- Or directly do multipartitioning using the other (3rd, 4th,...) singular vectors.

# Generic Summarization and Keyphrase Extraction Using Mutual Reinforcement Principle and Sentence Clustering

paper by Hongyuan Zha

# Extractive Summarization

- Based on selecting most “important” sentences in a set of documents.
- Redundancy vs. coverage
  - First cluster sentences into subtopics.
  - And then select the most important sentence from each cluster.

# Mutual Reinforcement Principle

- *“A term should have a high saliency score if it appears in many sentences with high saliency scores while a sentence should have a high saliency score if it contains many terms with high saliency scores.”*
- Same bipartite representation of terms and documents:
  - $W$  is the  $m \times n$  (terms x documents) matrix
  - Edge weights  $w_{ij}$  could be *tf* or *tf-idf*

# Mutual Reinforcement Principle

- Mutual reinforcement principle in matrix form:

$$u = \frac{1}{\sigma} W v, \quad v = \frac{1}{\sigma} W^T u,$$

- The Algorithm:

- Initialize  $v = [1 \ 1 \ \dots \ 1]$

- Repeat until convergence:

- Compute and normalize

$$u = W v, \quad u = u / \|u\|,$$

- Compute and normalize

$$v = W^T u, \quad v = v / \|v\|$$

- Suppose we have a prior weight distribution  $D_s$  for sentences. Then use  $W D_s$  instead of  $W$ .

# Sentence Clustering

- Consider an *undirected weighted* graph  $W_s$  of sentences only.
- The edge weights  $w_{ij}$  are the similarities (e.g. cosine) between the documents.
- Make the links between near-by (consecutive) sentences by adding a constant weight ( $w_{ij} + \alpha$ )
- $\alpha$  can be tuned by cross validation.

# Sentence Clustering

- The resultant sentence similarity matrix:  $W_s(\alpha)$
- Use spectral relaxation of the k-means algorithm on this matrix to cluster the sentences. Details are in the paper.
- Optionally use  $\frac{\log_2(\text{number of terms in sentence} + 1)}{\log_2(i + \text{total number of sentences})}$  for sentence priors.

# Experiments

- Sentence clustering is evaluated against the section structure of the documents.
- Clustering results with estimated  $\alpha$  are close to the results with optimal  $\alpha$  values.
- No quantitative evaluation for sentence extraction.

# Spectral Learning

paper by Kamvar, Klein & Manning

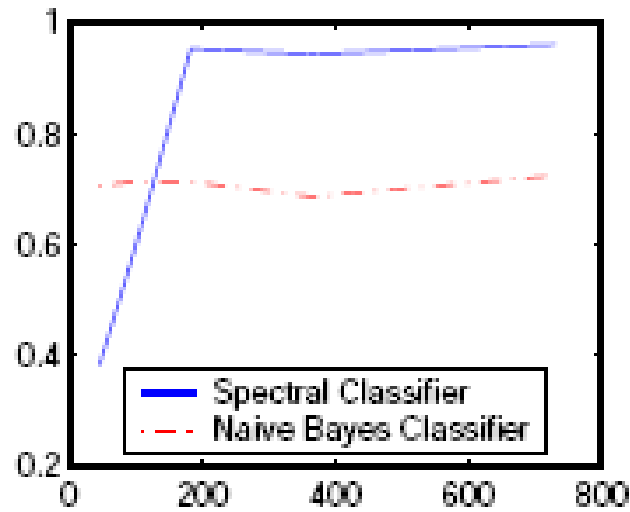
# Spectral Clustering

- Form the affinity matrix  $A$ .
- Let  $D$  be the degree (row sums) matrix.  
Construct  $L = D^{-1/2}AD^{-1/2}$  (or  $D^{-1}A$ )
- Find  $x_1, x_2, \dots, x_k$ ,  $k$  largest eigenvectors of  $L$ . Form the matrix  $X = [x_1 \ x_2 \ \dots \ x_k]$  by stacking the eigenvectors in columns.
- Normalize the rows of  $X$  to be unit length.
- Treating each row of  $X$  as a data point, cluster them using some clustering algorithm (e.g  $k$ -means)

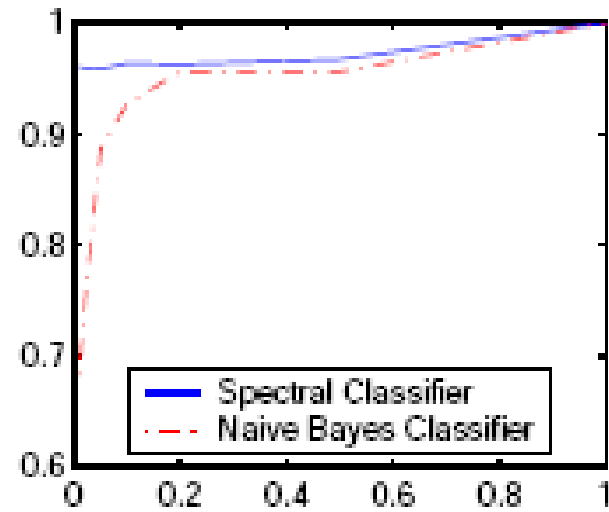
# Spectral Classification

- Connect the nodes that are in the same class with edge weight 1.
- Do not connect the nodes that are in different classes.
- Do the same steps. Finally use a classification algorithm (nearest neighbor, Naïve Bayes, etc.) for the rows of  $X$ .

3 news:

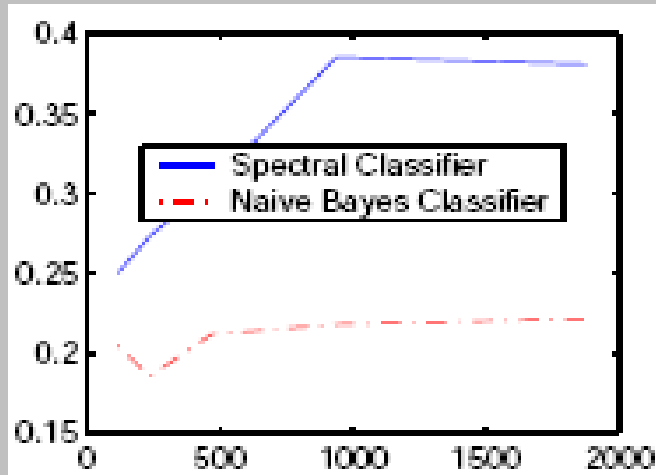


(a) Adding Unlabeled Data

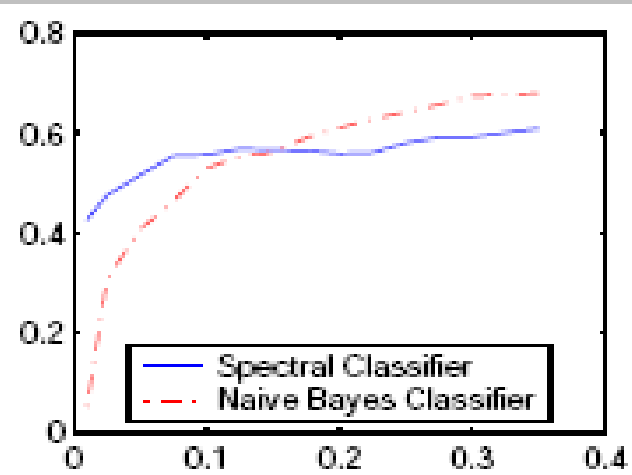


(b) Adding Labeled Data

20 news:



(a) Adding Unlabeled Data



(b) Adding Labeled Data