Using Spectral Clustering for Author Name Disambiguation

Zhou Xuanyi
School of Electrical and
Computer Engineering
Shanghai Jiao Tong University
Email: zxy1998@sjtu.edu.cn

***Abstract*—Author name disambiguation has long been a problem in bibliography websites. In AceMap, a hierarchical clustering method is originally used to tackle this problem. Although the method achieves decent performance, it has several drawbacks. In this artical we present a method based on spectral clustering that achieves similar accuracy but requires less computational resources.**

I. Introduction

AceMap is a academic map that collects scholars' information for indexing and querying. It uses network spiders to retrieve information from the internet, usually in the format of plain text, then further processes the information to construct a dataset that's suitable for querying. With the development of the system, the AceMap group has accumulated a relatively large amount of data. However, one major problem faced when processing the data is that there could be different people sharing the same name. Had two people been treated as the same person simply because their names are identical, the reliability and correctness of the database would be harmed greately. Author name disambiguation aims to solve this problem.

The disambiguation process aims to tell if articles, of which one or more authors have identical names, are published by the same person, with the aid of additional information about the publications. Typically, there are two major types of methods to achieve this goal: author grouping methods and author assignment methods. Author grouping methods use clustering algorithms to classify publications, while author assignment methods try to assign publications to known authors. In this paper, the focus is primarily on author grouping methods.

Originally, the method used in AceMap is based on hierarchical clustering. Although it achieves a relatively high accuracy, it has several drawbacks such as being too slow on large datasets, being too complicated, and relying on a large number of handcrafted parameters. The proposed method is largely free of such drawbacks at the cost of reduced accuracy. The two methods are compared in detail in section VI: experiments.

II. Problem Formulation

Given a set of n publications $P = \{p_1, p_2, \ldots, p_n\}$, the goal is to partition them into k sets $S = \{P_1, P_2, \ldots, P_k\}$ such that $P_1 \cup P_2 \cup \cdots \cup P_k = P$ and $P_i \cap P_j = \varnothing$ if $i \neq j$, where each set contains the publications of and only of one author.

A *publication* contains information about an document. Ferreira et al. [2] classify the information into the following three categories: citation infrmation, which are attributes directly extracted from the publications such as title, author names, year, and so on; web information, which are retrieved from the Internet, usually by searching on search engines; and implicit evidence, which are implicit attributes inferred from visible ones. Combined usage of multiple types of attributes can greatly help the disambiguation process and improve accuracy.

III. Related Works

In [2], current methods are categorized into two classes: author grouping methods and author assignment methods.

A. Author Grouping Methods

In author grouping methods, a clustering method is used to classify each publication. Han et al. [3] represents each publication as a feature vector before using K-way spectral clustering with QR decomposition [10] to cluster the publications. The number of clusters needs to be informed in this method.

B. Author Assignment Methods

In author assignment methods, a model of each author is constructed, and each publication is directly assigned to an author. Such methods can be further divided into two types: classification and clustering.

1) Classification: A supervised machine learning technique is used to assign publications to authors. In [8], Veloso et al. proposed a disambiguation method that uses a supervised rule-based associative classifier for author assignment.

2) Clustering: Such methods optimizes the fit between a set of publications and some mathematical model used to represent that author. In [7], Tang et al. proposed a probabilitic framework based on Hidden Markov Random Models (HMRF) for this problem.

IV. Method Previously Used in AceMap

The method previously used in AceMap is based on hierarchical clustering, as illustrated in Algorithm 1. The algorithm models the publications and the relationships between them as a heterograph, in which the nodes are publications, and edges between them are split into two classes, strong edges and weak edges, based on the affiliation, published venue, and author names of two publications. More specifically, if two of these attributes (author names are counted separately, and the name of the target author is excluded) are the same, then a strong edge is established between two nodes. If only one of these attributes of the two publications are the same, a weak edge is established instead. The algorithm first merges all papers that have strong connections between them, then uses hierarchical clustering to further cluster the publications according to the similarity matrix. Finally, the algorithm tries to merge scattered publications, i.e., publications that are the only elements in their clusters, into other larger clusters.

A. Building the Similarity Matrix

After computing all weak edges, it is easy to obtain the types of these edges that indicate which attributes of two publications are the same. Since all node connected by strong edges have already been merged into clusters, there may exist multiple weak edges of different types between two clusters. A weight is assigned to each edge based on its type and the similarity between the titles of the two publications it connects, then the Dijkstra algorithm is used to find the shortest distance between every two clusters. Specifically, for each cluster c and all its neighbors to which it's connected by an edge of type t, the shortest path algorithm tries to find a path to each neighbor that consists of no edges of type t, therefore finding two types of links between the two clusters. After computing the shortest distances between every two clusters for each link type, these distances are then assigned to each pair of nodes that are connected by a weak edge of the corresponding type and are not in the same cluster.

B. Hierarchical Clustering

After the similarity matrix has been computed, the hierarchical clustering process tries to further cluster the nodes by repeatedly merging the two nodes that are not in the same cluster and are connected by the shortest edge. In the original implementation, however, the node pairs are further examined to make sure that the two clusters don't have conflicts with one another. The conditions are controlled by constants K_1, K_2, and K_3. Firstly, the total number of weak edges that connect the two clusters must be greater than K_1 times the product of the two clusters' cardinalities. Secondly, The total number of different affiliations of the combined cluster must not exceed K_2, limiting the number of places that one may have worked at. Finally, let Y be the set of all papers published in the same year, then the number of different affiliations of these papers must not exceed K_3. This limits the frequency one changes to work at different places. In the original implementation, $K_1 = 0.01$, $K_2 = 20$, and $K_3 = 3$.

C. Merging Scattered Publications

During this phase, the algorithm tries to merge clusters that have only one publication in each of them with larger clusters based primarily on the similarity between their titles. This process is controlled by K_4, K_5, and K_6. For each scattered paper, the algorithm iterates through all other papers in descending order of the similarities between their titles, trying to merge the paper with each of them. For two papers whose titles' similarity is s, they are not merged if $s < K_4$, or if $s < K_5$ when the two papers are neighbors. Suppose the other paper belongs to cluster c, then the two clusters are not merged if the number of elements in c is smaller than K_6. Finally, a relaxed version of conflict checking in the hierarchical clustering phase is used to ensure that the scattered paper doesn't conflict with any paper in the other cluster. The first part of conflict checking is ignored in this phase, so there's no requirements of the number of edges between the two clusters. In the original implementation, $K_4 = 0.8$, $K_5 = 0.6$, and $K_6 = 5$.

V. Spectral Clustering

Donath and Hoffman [1] first suggested constructing graph partitions based on eigenvectors of the adjacency matrix. The spectral clustering algorithm was then discovered, re-discovered, and extended many times in different communities [6]. In 2000, Shi and Malik [5] proposed normalized spectral clustering and sparked its popularity in the machine learning community.

The spectral clustering algorithm aims to find a partition of a similarity graph such that edges between two different groups have small weights while edges within a group has large weights. Given a set of data points $x_1, \dots, x_n$ and a similarity metric $s(x_1, x_2) \geq 0$, the *similarity graph* $G = (V, E)$ represents the data based on the similarities between data points, in which two vertices x_1 and x_2 are connected if $s(x_1, x_2)$ is larger than zero or a certain threshold. The edge between two connected vertices is weighted also by $s(x_1, x_2)$. The weighted *adjacency matrix* of the graph is the matrix $W = (w_{ij})_{i,j=1,\dots,n}$, where $w_{ij} = w_{ji}$. If $w_{ij} = 0$, then the vertices v_i and v_j are not connected. The degree of a vertex $v \in V$ is defined as

$$d_i = \sum_{j=1}^{n} w_{ij},$$

and the *degree matrix* D is defined as the diagonal matrix with the degrees $d_1, \dots, d_n$ on its diagonal. Two major

Algorithm 1: DISAMBIGUATION-ACEMAP

```
Data: The set of publications $P$, and parameters $K_1, \ldots, K_6$
Result: The set of clusters $C$

1  Procedure Shortest-Path($t$, $C$, $s$)
2      foreach $c \in C$ do
3          $Dist[c] \leftarrow \infty$;
4      $Dist[s] \leftarrow 0$;
5      Compute the weights of all edges based on their
         types and the similarities between the titles of
         the two publications that an edge connects;
6      Use Dijkstra algorithm to find the shortest path
         from $s$ to every other cluser, taking into
         consideration only edges whose types are *not* $t$,
         and store the shortest distances in $Dist$;
7      return $Dist$;

8  Procedure Neighbors($C$, $ClusterEdge$, $n$)
9      $S \leftarrow \{\}$;
10     foreach $\{c_1, c_2, t\} \in ClusterEdge$ do
11         if $c_1 = n$ then
12             $S[t] \leftarrow S[t] \cup \{c_2\}$;
13         else if $c_2 = n$ then
14             $S[t] \leftarrow S[t] \cup \{c_1\}$;
15     return $S$;

16 Procedure No-Conflict($c_1$, $c_2$, $NodeWeights$,
     $strict$)
17     if $strict$ then
18         $N \leftarrow 0$;
19         foreach $\{p_1, p_2, w\} \in NodeWeights$ do
20             if $p_1 \in c_1$ and $p_2 \in c_2$ then
21                 $N \leftarrow N + 1$;
22         if $N < K_1 \cdot |c_1| \cdot |c_2|$ then
23             return FALSE;
24     $A_1 \leftarrow \{\text{AFFILIATION}[p] \mid p \in c_1\}$;
25     $A_2 \leftarrow \{\text{AFFILIATION}[p] \mid p \in c_2\}$;
26     if $|A_1 \cup A_2| > K_2$ then
27         return FALSE;
28     foreach $y \in \{\text{YEAR}[p] \mid p \in c_1 \cup c_2\}$ do
29         $Y_1 \leftarrow \{p \mid p \in c_1, \text{YEAR}[p] = y\}$;
30         $Y_2 \leftarrow \{p \mid p \in c_2, \text{YEAR}[p] = y\}$;
31         if $|Y_1 \cup Y_2| > K_3$ then
32             return FALSE;
33     return TRUE;

34 $C \leftarrow \{\{p\} \mid p \in P\}$;
35 foreach $\{p_1, p_2\} \subset P$, where $p_1 \neq p_2$ do
36     $F_{\text{aff}} \leftarrow \text{AFFILIATION}[p_1] = \text{AFFILIATION}[p_2]$;
37     $F_{\text{venue}} \leftarrow \text{VENUE}[p_1] = \text{VENUE}[p_2]$;
38     $N_{\text{co-auth}} \leftarrow |\text{AUTHORS}[p_1] \cap \text{AUTHORS}[p_2]| - 1$;
39     $S \leftarrow N_{\text{co-auth}} + F_{\text{aff}} + F_{\text{venue}}$;
40     if $S \geq 2$ then
41         Merge(Cluster-Of($p_1$), Cluster-Of($p_2$));
42     else
43         $NodeEdge[p_1][p_2] \leftarrow$ Source($S$);
44 foreach $\{p_1, p_2, t\} \in NodeEdge$ do
45     $c_1 \leftarrow$ Cluster-Of($p_1$);
46     $c_2 \leftarrow$ Cluster-Of($p_2$);
47     if $c_1 \neq c_2$ then
48         Add($ClusterEdge[c_1][c_2][t]$, $\{p_1, p_2\}$);
49 foreach $c \in C$ do
50     foreach $\{t, C'\} \in$ Neighbors($C$, $ClusterEdge$,
         $c$) do
51         $Dist \leftarrow$ Shortest-Path($t$, $C$, $c$);
52         foreach $c' \in C'$ do
53             if $c' \neq c$ and $Dist[c'] \neq \infty$ then
54                 $ClusterWeights[c][c'][t] \leftarrow Dist[c']$;
55 foreach $\{p_1, p_2, t\} \in NodeEdge$ do
56     $c_1 \leftarrow$ Cluster-Of($p_1$);
57     $c_2 \leftarrow$ Cluster-Of($p_2$);
58     $NodeWeights[p_1][p_2] \leftarrow ClusterWeights[c_1][c_2][t]$;
59 foreach $\{p_1, p_2, w\} \in$ Sorted($NodeWeights$) do
60     $c_1 \leftarrow$ Cluster-Of($p_1$);
61     $c_2 \leftarrow$ Cluster-Of($p_2$);
62     if No-Conflict($c_1$, $c_2$, $NodeWeights$, TRUE)
         then
63         Merge($c_1$, $c_2$);
64 foreach $c \in C, |c| = 1$ do
65     $p \leftarrow$ The element in $c$;
66     $P' \leftarrow P$ ordered by Similarity(TITLE[$p$],
         TITLE[$p'$]);
67     foreach $p' \in P'$ do
68         $s \leftarrow$ Similarity(TITLE[$p$], TITLE[$p'$]);
69         if $s > K_4$ or ($s > K_5$ and $p$ is a neighbor of
             $p'$) then
70             $c' \leftarrow$ Cluster-Of($p'$);
71             if $|c'| > K_6$ and No-Conflict($c$, $c'$,
                 $NodeWeights$, FALSE) then
72                 Merge($c$, $c'$);
73 return $C$;
```

types of criteria for good partitions, RatioCut and Ncut, have objective functions are defined as

$$\text{RatioCut}(A_1, \ldots, A_k) = \sum_{i=1}^{k} \frac{\text{cut}(A_i, \overline{A}_i)}{|A_i|},$$

$$\text{Ncut}(A_1, \ldots, A_k) = \sum_{i=1}^{k} \frac{\text{cut}(A_i, \overline{A}_i)}{\text{vol}(A_i)},$$

where each A_i represents a partition and vol(A) is the sum of the degrees of all vertices in the subgraph A. The *unnormalized spectral clustering* algorithm seeks to minimize RatioCut, while the *normalized spectral clustering* algorithm seeks to minimize Ncut.

Both unnormalized and normalized spectral clustering operate according to Algorithm 2, and the only difference is in the `Laplacian()` function. Specifically,

$$\texttt{Laplacian}(W) = \begin{cases} D - W & \text{unnormalized} \\ I - D^{-1}W & \text{normalized} \end{cases}$$

As can be seen from the pseudo-code, computing the Laplacian matrix and its eigenvalues and eigenvectors is essentially an embedding process. The resulting vectors are the rows of the matrix formed by combining the eigenvectors corresponding to the k smallest eigenvalues. The K-means algorithm is then invoked to cluster the embedded vectors.

Algorithm 2: Spectral-Clustering

```
Data: The adjacency matrix $W$, and the number
      of clusters $k$
Result: The labels of each node $R$

1 $L \leftarrow$ Laplacian($W$);
2 $\{(\lambda_1, t_1), \ldots, (\lambda_n, t_n)\} \leftarrow$
    Eigen-Values-And-Vectors($L$);
3 $t_1, \ldots, t_n \leftarrow$ The eigenvectors in ascending order of
    the eigenvalues;
4 $\begin{bmatrix} p_1 & p_2 & \cdots & p_n \end{bmatrix} \leftarrow \begin{bmatrix} t_1 & t_2 & \cdots & t_k \end{bmatrix}^\top$;
5 $R \leftarrow$ K-Means($p_1$, $p_2$, ..., $p_n$);
6 return $R$;
```

A. Usage in Author Disambiguation

In the proposed disambiguation algorithm, a similarity graph is built before spectral clustering is used to classify the publications. The similarity graph is built in a way very similar to how the similarity matrix is obtained in the original method. For every two publications, the number of identical attributes of them are calculated as in section IV-A, and a "strong" edge is added between any pair of papers if two of their attributes are the same. Parameter K_1 is used to control the minimum number of outgoing edges that a node has, and if a node has less than K_1 outgoing edges after this step, more edges are added based on the similarities of the papers' titles. Specifically, if a paper p has $k < K_1$ outgoing edges, all other papers are ordered by the similarity between the title of p and that of the other paper. For the first $K_1 - k$ papers with the largst similarities, they are linked with p with either a "weak" edge if there's an attribute that's the same between the two papers, or otherwise a "disconnected" edge. An alternate strategy is to only consider papers with similarities larger than a threshold, but in practice it produces less accurate results. After the edges are added, the similarity matrix is generated, of which each element is the corresponding edge's weight. A "strong" edge always has weight 1, while "weak" edges and "disconnected" edges have weights K_2 and K_3, respectively. The similarity matrix is then used for unnormalized spectral clustering. In the implementation, $K_1 = 5$, $K_2 = 0.3$, and $K_3 = 0.1$. See Algorithm 3 for the pseudocode for building the similarity graph.

Algorithm 3: Generate-Similarity-Graph

```
Data: The set of publications $P$, and the
      parameters $K_1$, $K_2$, and $K_3$
Result: The similarity matrix $W$

1  $N \leftarrow |P|$;
2  $W \leftarrow [0]_{N \times N}$;
3  foreach $p \in P$ do
4      $T \leftarrow \{\}$;
5      foreach $p' \in P$ do
6          $F_{\text{aff}} \leftarrow \text{AFFILIATION}[p] = \text{AFFILIATION}[p']$;
7          $F_{\text{venue}} \leftarrow \text{VENUE}[p] = \text{VENUE}[p']$;
8          $N_{\text{co-auth}} \leftarrow |\text{AUTHORS}[p] \cap \text{AUTHORS}[p']| - 1$;
9          $S \leftarrow N_{\text{co-auth}} + F_{\text{aff}} + F_{\text{venue}}$;
10         if $S > 1$ then
11             $W[p][p'] \leftarrow 1$;
12         else
13             $T \leftarrow T \cup \{p'\}$;
14     $k \leftarrow$ Num-Neighbors($p$);
15     if $k < K_1$ then
16         $P' \leftarrow P$ sorted descending by the
             similarities of titles, taking only the first
             $K_1 - k$ elements;
17         foreach $p' \in P'$ do
18             if $p' \in T$ then
19                 $W[p][p'] \leftarrow K_2$;
20             else
21                 $W[p][p'] \leftarrow K_3$;
22 return $\frac{1}{2}(W + W^\top)$;
```

VI. Experiments

In the experiments, the AMiner disambiguation dataset [9] is used. The dataset provides the title, author

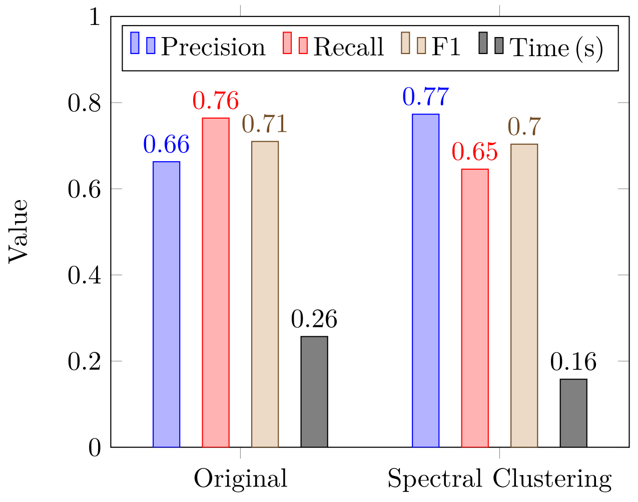

Fig. 1. Comparing the performance of the two methods

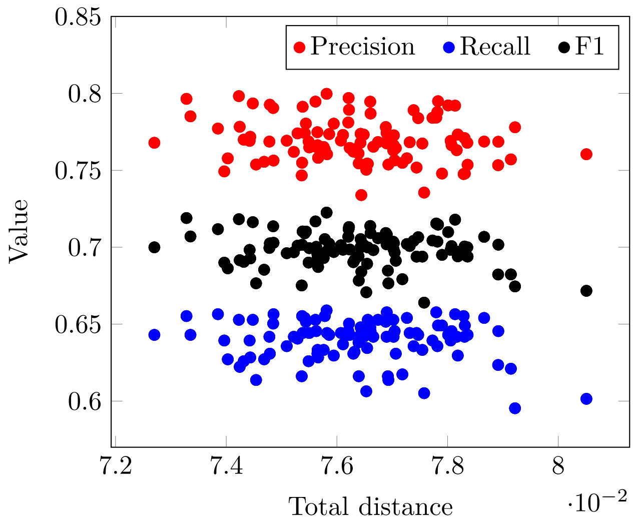

Fig. 2. Relationship between the sum of distance and accuracy

names, year, publication venue title, and the target author's affiliation, of each publication.

A. Performance

The accuracy and running time of the two algorithms are compared in Fig. 1. As can be seen from the figure, spectral clustering has higher precision but lower recall, resulting in a F1 score similar to the original method. However, the proposed method reduces the computing time by approximately 40%, and therefore is more efficient. During the experiments, it was observed that unnormalized spectral clustering performs far better than normalized spectral clustering, and adding thresholds to similarities is usually a detriment to the performance as well.

B. Randomness

An interesting trait of the K-means algorithm is that for different initial centroids ("seeds"), the algorithm may converge at different local minimums. A useful metric for evaluating the results of the algorithm is the sum of distances between each data point and the centroid of the cluster it belongs to. In the second experiment, the K-means algorithm is ran a large number of times on the same embedding with different seeds, and the accuracy is calculated for each result. The results shows that, unfortunately, the sum of distances is not very useful for obtaining a better result since there's very little correlation between the sum of distances and accuracy, as depicted in Fig. 2.

C. Embedding

As mentioned before, the proposed algorithm obtains an embedding of the data points. In Fig. 3, PCA [4] is used to lower the dimensionality of the embedding for visualization, and data points are colored according to the clusters they belong to. As can be seen in the figure, although the embedding yields decent clustering results,

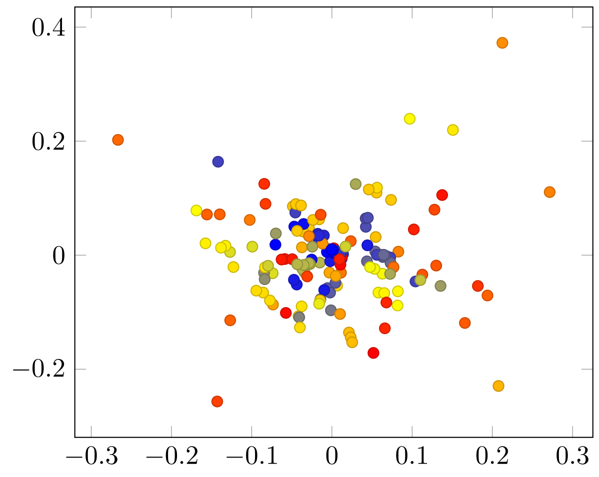

Fig. 3. Embedded publications

it's not suitable for visualization when processed with PCA.

VII. Comparison of the Two Methods

The spectral clustering algorithm achieves an accuracy similar to that of the previous method based on hierarchical clustering, but requires less running time. It had higher precision but lower recall in the experiments, and was also observed to tend to generate clusters whose sizes are more balanced. The proposed method is also much simpler and requires less handcrafted parameters. While the original method is a deterministic method that's bound to return the same result for the same input, the proposed method can return different results due to the randomly generated seeds in the K-means algorithm, which can be both a blessing and a curse. Due to this characteristic, one can run the algorithm multiple times in the hope of obtaining a better result, but unfortunately no good metric is found yet that can evaluate a partition without ground truth labels. Finally, the proposed method can also be used to obtain an embedding of the publications.

One major disadvantage of the method is that it requires that the number of clusters be known before clustering. Although it can also be determined by maximizing the eigengap $|\lambda_{k+1} - \lambda_k|$, the result is often inferior. Moreover, since the information of publications are compacted into the similarity graph, a lot of information that can be used to fine-tune the clusters are lost, and a process similar to the "merge scattered papers" phase of the original method or the conflict detection process, is not easily made possible. In fact, the "published year" attribute of papers is left completely unused in the implementation.

VIII. Conclusion and Future Work

We presented an author disambiguation method based on spectral clustering has similar accuracy with the original method in AceMap based on hierarchical clustering, but is more compact and faster by about 40%. Unlike the original method which is deterministic, the results of the new method is affected by the choice of seeds of the K-means algorithm. The proposed method can also be used to obtain an embedding of the publications.

While the method has many advantages, there are still many aspects in which it can be improved. Firstly, the current way of automatically finding the number of clusters is not very reliable. Secondly, the generated embedding cannot be simply processed by PCA for visualization. Finally, it would improve the algorithm's accuracy greatly if more information could be represented in the similarity graph.

References

[1] William E Donath and Alan J Hoffman. Lower bounds for the partitioning of graphs. *IBM Journal of Research and Development*, 17(5):420–425, 1973.

[2] Anderson A Ferreira, Marcos André Gonçalves, and Alberto HF Laender. A brief survey of automatic methods for author name disambiguation. *Acm Sigmod Record*, 41(2):15–26, 2012.

[3] C Lee Giles, Hongyuan Zha, and Hui Han. Name disambiguation in author citations using a k-way spectral clustering method. In *Digital Libraries, 2005. JCDL'05. Proceedings of the 5th ACM/IEEE-CS Joint Conference on*, pages 334–343. IEEE, 2005.

[4] LIII KPFRS. On lines and planes of closest fit to systems of points in space. In *Proceedings of the 17th ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems (SIGMOD)*, 1901.

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

[6] Daniel A Spielmat and Shang-Hua Teng. Spectral partitioning works: Planar graphs and finite element meshes. In *Foundations of Computer Science, 1996. Proceedings., 37th Annual Symposium on*, pages 96–105. IEEE, 1996.

[7] Jie Tang, Alvis CM Fong, Bo Wang, and Jing Zhang. A unified probabilistic framework for name disambiguation in digital library. *IEEE Transactions on Knowledge and Data Engineering*, 24(6):975–987, 2012.

[8] Adriano Veloso, Anderson A Ferreira, Marcos André Gonçalves, Alberto HF Laender, and Wagner Meira Jr. Cost-effective on-demand associative author name disambiguation. *Information Processing & Management*, 48(4):680–697, 2012.

[9] Xuezhi Wang, Jie Tang, Hong Cheng, and S Yu Philip. Adana: Active name disambiguation. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, pages 794–803. IEEE, 2011.

[10] Hongyuan Zha, Xiaofeng He, Chris Ding, Ming Gu, and Horst D Simon. Spectral relaxation for k-means clustering. In *Advances in neural information processing systems*, pages 1057–1064, 2002.