Scalable Name Disambiguation using Multi-level Graph Partition

Byung-Won On
Penn State University, USA
on@cse.psu.edu

Dongwon Lee
Penn State University, USA
dongwon@psu.edu

Abstract

When non-unique values are used as the identifier of entities, due to their homonym, confusion can occur. In particular, when (part of) "names" of entities are used as their identifier, the problem is often referred to as the *name disambiguation* problem, where goal is to sort out the erroneous entities due to name homonyms (e.g., if only last name is used as the identifier, one cannot distinguish "Vannevar Bush" from "George Bush").

In this paper, in particular, we study the scalability issue of the name disambiguation problem – when (1) a small number of entities *with large contents* or (2) *a large number of* entities get un-distinguishable due to homonyms, how to resolve it? We first carefully examine two of the state-of-the-art solutions to the name disambiguation problem, and point out their limitations with respect to scalability. Then, we adapt the *multi-level graph partition* technique to solve the large-scale name disambiguation problem. Our claim is empirically validated via experimentation – our proposal shows orders of magnitude improvement in terms of performance while maintaining equivalent or reasonable accuracy compared to competing solutions.

1 Introduction

In many applications, entities need to carry a unique identifier. This identifier can be as simple as a primary key in Databases or ISBN of books, or as complex as DNA fingerprint of people. When all applications adopt universal identifier system such as DOI, one does not have to worry about issues related with identifiers. However, in reality, it is not uncommon to find an application that uses non-unique data values as an identifier. One of commonly used such identifiers is a short *name description* ("names" in short hereafter) of entities. Examples include: name of persons, name of movies, name of cities, etc. Since these names of entities are not unique, inevitably, there can be multiple entities with the same names, causing a confusion. This problem is often referred to as the **Name Disambiguation** problem, where goal is to sort out the erroneous entities due to name homonyms.

In general, one can model the name disambiguation problem as *k-way clustering* problem. That is, given a set of mixed N entities with the same name description d, group N entities into k clusters such that entities within each cluster belong to the same real-world group (e.g., same author or movie).

In this paper, in particular, we study the scalability issue of the name disambiguation problem – when a large number of entities get un-distinguishable due to homonyms, how to resolve it? By and large, the scalability issue has been ignored in previous research of name disambiguation problem. Therefore, existing solutions tend to work well for a handful of mixed entities in the range of 10 or so, or a large number of entities with limited number of feature dimensions (e.g., [9, 2]). However, as data applications become more complicated and users increase rapidly, new needs arise to handle more large-scale name disambiguation problem. For instance, for a given "name" query keyword t (e.g., person, company, or movie), it is common to have thousands of web pages returned from search engines, all of which contain the keyword t and could have high dimension spaces in the vector space model. Therefore, it is important to have a scalable yet accurate name disambiguation algorithm.

For this goal, in this paper, we first carefully examine two of the state-of-the-art solutions – k-way spectral clustering [9] and multi-way distributional clustering [2] – to the name disambiguation problem, and point out their limitations with respect to their scalability. Then, we adapt the *multi-level graph partition* technique to solve the large-scale name disambiguation problem. Our claim is empirically validated via experimentation – our proposal shows orders of magnitude improvement in terms of performance while maintaining equivalent or reasonable accuracy.

2 Overview of Problem & Solution

Formally, using the terms of Table 1, the name disambiguation problem in our setting is defined as follows:

Name	Description
k	# of clusters
l	# of tokens
m	# of unique tokens (i.e., m-dimensional vectors)
n	# of documents (i.e., entities)
c	Constant

Table 1: Terms.

Given a set of mixed entities E=$\{e_1, ..., e_n\}$ with the same name description d, group E into k disjoint clusters C=$\{c_1, ..., c_k\}$ ($k \leq n \leq m \leq l$) such that entities $\{e_p^i, ..., e_q^i\}$ ($1 \leq p \leq q \leq n$) within each cluster c_i belongs to the same real-world group.

Note that both n and k can be a substantially large number. In general, distance functions to measure the distance between two entities in the name disambiguation problem is more expensive than those used in conventional clustering framework. This is because each entity can be a long record or a whole document, instead of simple numeric or string values of attributes.

The basic three-phased framework of our approach is as follows:

- We first represent all entities and their inter-relationships as a graph $G = (V, W)$. For instance, each node $v(\in V)$ represents an entity $e_i(\in E)$ and each edge $(v_1, v_2)(\in W)$ carries a non-negative weight to capture the relationship between v_1 and v_2;
- Second, the original graph G is successively condensed into smaller graphs G_1, ..., G_x such that $|V| > |V_1| > ... > |V_x|$ where $|V_i|$ is the number of vertices in the graph G_i and $i \in [1..x]$; and
- Third, the condensed graphs are reversely uncoarsened. Suppose that a graph G_i is condensed into a graph G_{i+1} in the second coarsening phase. If vertices v_1 and v_2 are bound to a single vertex (v_1, v_2), and then the vertex (v_1, v_2) belongs to a cluster c in level $i + 1$, then the vertex v_1 and v_2 will be grouped to the same cluster c in the uncoarsening phase.

3 Two State-of-the-art Solutions: MDC & SC

3.1 Multi-way Distributional Clustering (MDC). Bekkerman and McCallum used the *multi-way distributional clustering (MDC)* to solve the name disambiguation problem in [2]. We briefly describe about MDC here.

Given a set of k clusters, $c_1, ..., c_k$, all tokens of $c_{i\in[1..k]}$ are placed in a *single* cluster while each entity $c_{i\in[1..k]}$ is placed in a *singleton* cluster. For instance, given a set of word tokens and documents in a collection, all the tokens are put in a single cluster while each document in the collection is assigned to each singleton cluster. Then, during top-down/bottom-up clustering iterations, the top-down clustering scheduler splits each element (e.g., a word) uniformly at random to two sub-clusters. On the other hand, the bottom-up clustering scheduler merges each element (e.g., a document in a collection) with its closest neighboring cluster. The scheduling scheme is pre-determined in MDC.

For example, if a schedule scheme is "words, words, words, documents, documents," three clustering iterations over words will be processed first, followed by two iterations over documents. Finally, for all elements, correct clusters are created based on Mutual Information – that is, it correctly clusters a random variable X (e.g., documents) by a joint probability distribution between X and an observed variable Y (e.g., words). The joint probability distribution is computed based on a table summarizing # of occurrences of times $x \in X$ occurred with $y \in Y$ (e.g., # of times a term y appears in a document x).

3.2 k-way Spectral Clustering (SC). Han et al. used the *k-way spectral clustering (SC)* to solve the name disambiguation problem in [9]. Here, we again briefly explain the SC.

Consider a set of entities E. The spectral clustering methods consider the similarity matrix S, where $S_{i,j}$ is a similarity measure between entities $e_p, e_q \in E$. As one of commonly used spectral clustering methods, *Shi-Malik* algorithm [17] partitions entities into two sets based on eigenvector v corresponding to the second smallest eigenvalue (i.e., Fiedler vector) of the Laplacian of S. Similarly, the *Meila-Shi* [14] and Han et al. [9] use the eigenvectors corresponding to k largest eigenvalues of the matrix $P = DS^{-1}$ for k, and then cluster entities using k-means or pivoted QR decomposition by their respective k components in eigenvectors.

3.3 Computational Complexity. The MDC method iteratively performs agglomerative clustering over terms (e.g., word tokens) and conglomerate clustering over documents (e.g., web pages or citations in a collection) at random, and assigns documents to more accurate clusters based on the joint probability distribution of terms and documents. Thus, this algorithm is significantly expensive on large-scale data.

For instance, suppose that the MDC method has two clustering systems X and Y. X is the agglomerative

clustering system over tokens such that a cluster $x \in X$ and an element $e_i \in X_{i=1..l}$. Y is the conglomerative clustering system over documents such that a cluster $y \in Y$ and an element $e_i \in Y_{i=1..n}$. Since the MDC method focuses on clustering documents, the maximal number of iterations to obtain the final clusters is $O(logn)$. During each iteration, each cluster in X is randomly split to two equally sized sub clusters and then each token $e_i \in x_i$ is correctly placed into x_j based on Mutual Information. Next, each cluster in Y is randomly merged to its nearest neighbor cluster and *cluster corrections* are performed to minimize the Bayes classification error. At each iteration in the top-down step, entity e_i is placed into a cluster x_j such that Mutual Information $I(X,Y)$ is maximal. Similarly, the same process is performed in the bottom-up step. Therefore, the computational complexity of MDC is:

$$O(l \cdot n \cdot logn)$$

On the other hand, in the k-way SC algorithm, given a similarity matrix M, $v_1, ...v_k$ eigenvectors of M are computed by the k largest eigenvalues to create the matrix V with k columns of eigenvectors of M. Finally, the rows of V are clustered. In general, the running time of these spectral clustering algorithms is dominated by the computation of eigenvectors of M, and the complexity time is known as [15, 10, 16, 6]:

$$O(\frac{4}{3} \cdot c \cdot m^3) \approx O(m^3)$$

As clearly shown here, since both MDC and SC have quadratic and cubic time complexities, they do not scale well. The *multi-level graph partition (MGP)* [11, 5] (to be elaborated in Section 4) consists of three steps. During the coarsening step, the size of the graph is repeatedly decreased; in the clustering step, the smallest graph is partitioned; and during the uncoarsening step, partitioning is successively refined to the larger graph. During the coarsening step, since the size of the graph is decreased from level to level and all the vertices in the graph are visited at each level, the complexity gets $O(\log n)$. In the clustering step, if we use one of spectral algorithms, the complexity is $O(\frac{4}{3} \cdot c \cdot \{20 \cdot k\}^3)$. During the uncoarsening step, the running time at each level is $O(nz)$, where nz is # of non-zero entries in the kernel matrix. Overall, therefore, the computational complexity of the MGP is:

$$O(\log n + \frac{4}{3} \cdot c \cdot (20 \cdot k)^3 + \log n) \approx O(k^3)$$

In conclusion, since $k \ll n$, the computational complexity of the MGP is the most efficient, compared to that of MDC and of SC.

4 MGP-based Name Disambiguation

In this section, we describe in details how to use MGP to solve the name disambiguation problem.

4.1 Graph Formation. We convert the given N entities into a graph $G = (V, E)$ as follows:

- Each entity e_i is mapped to a node $v_i (\in V)$.
- By treating the contents of an entity e_i as documents, we apply the standard vector space model to convert e_i into an m–dimensional vector (e.g., $X = (\alpha_1, ..., \alpha_m)$). If the i–th token in the entire token space appears in an entity e_i, then α_i is the TF/IDF weight value of the i's token. Otherwise, $\alpha_i = 0$.
- Finally, edge weight between two entities x and y is computed as follows [3]: $TFIDF(x,y) = \sum_{w \in T_x \cap T_y} V(w, T_x) \times V(w, T_y)$, where (1) $V(w, T_x) = log(TF_{w,T_y} + 1) \times \frac{log(IDF_w)}{\sqrt{\sum_{w'} (log(TF_{w,T_y}+1) \times log(IDF_w))}}$ (symmetrical for $V(w, T_y)$), and (2) $V(w, T) = log(TF_{w,T} + 1) \times \frac{log(IDF_w)}{\sqrt{\sum_{w'} (log(TF_{w,T}+1) \times log(IDF_w))}}$. In the TFIDF similarity function, TF_{w,T_x} is the frequency of w in T_x, and IDF_w is the inverse of the fraction of names in a corpus containing w;

4.2 Multi-level Graph Partition. For a multi-level graph partitioning algorithm that is desired in our domain, we use Dhillon et al. [5]. Formally, the multi-level graph partitioning algorithm works as follows:

- **The coarsening phase.** The original graph G is successively subdivided into smaller graphs $G_1, G_2, ..., G_k$ such that $|V| > |V_1| > ... > |V_k|$, where level $i \in [1..k]$ and $|V_i|$ is the number of vertices in graph G_i. In G_i, visit each vertex randomly, and then merge a vertex v with a neighbor w that maximizes the edge weight between v and w. Once all the vertices in G_i are visited, the coarsening process at level i is completed.
- **The partitioning phase.** Through the repeated coarsening steps, the smallest graph is determined such that the size of the graph is less than $20 \times k$. Then, the spectral algorithm of Yu and Shi [21] is performed to cluster the smallest graph.
- **The uncoarsening phase.** In order to derive partitions of the original graph G, the uncoarsening step is required in which the clustered graphs are repeatedly projected to larger graph. Suppose that the size of G_i was decreased into that of G_{i+1} in

the coarsening phase. Furthermore, two vertices v and w of G_i were binded to a single vertex (v, w) of G_{i+1}. Then, the vertex (v, w) was partitioned into a cluster c in the partitioning phase. In the uncoarsening phase, the vertices v and w can be grouped to the same cluster c. Then, more accurate projection to the larger graph is performed using a weighted kernel k-means uncoarsening algorithm. According to [5], graph clustering objective functions (e.g., Ratio Association) can be transformed into weighted kernel k–means objectives, with the weight of vertex w and kernel matrix K, to locally optimize these graph clustering objectives. Therefore, given a similarity matrix M, the kernel matrix K of a graph clustering objective function (i.e., Ratio Association) is computed by $\sigma I + M$ where σ is a real number. Subsequently, compute the updated clusters as $argmin_k(K_{xx} - \frac{2\times\Sigma_{y\in\pi_k^i}\times w_y\times K_{xy}}{\Sigma_{y\in\pi_k^i}\times w_y} + \frac{\Sigma_{y,z\in\pi_k^i}\times w_y\times w_z\times K_{yz}}{(\Sigma_{y\in\pi_k^i}\times w_y)^2})$, where π_k^i is the k-th cluster in the i-th iteration.

5 Experimental Validation

For the MDC, k-way SC, and MGP methods, we used the implementation of [1], [20], and [4], respectively. For the implementation of TF/IDF Cosine similarity, we used SecondString [19]. All experimentation were done on 4 × 2.6Ghz Opteron processors with 32GB of RAM.

5.1 Set-up. For validation, we have used four data sets – two small and two large data sets from real examples. Table 2 illustrates the overall statistics of four datasets.

- First, the `ACM-s` is real test case that we have gathered from the ACM digital library. When two scholars have the same name spellings, their citation data are mistakenly merged into a single collection, leading to an incorrect citation analysis results. We collected 24 real examples, and manually checked their correctness.
- The `WWW-s` is a small-sized test case using the 1,085 web pages that [2] used. In 2004, [2] extracted 12 personal names from Melinda Gervasio's email directory. Then, 100 top-ranked web pages of each name were retrieved from Google, and cleaned and manually labeled by authors. The resulting dataset consists of 1,085 web pages, 187 different persons, and 420 relevant pages. For the details, please refer to [2].
- Next, the `DBLP-m` is a medium-sized citation test case generated from DBLP digital library. To generate an ambiguous name dataset, we clustered author names from the entire DBLP citation data. last name. Then, we sorted the formed name clusters by the number of name variants. Finally, we obtained top-10 ambiguous names.
- Finally, `DBLP-l` is a large-scale citation test case, similar to `DBLP-m`, except that this time only the full last name is used in the blocking.

Test Case	k	n	m
`ACM-s`	2	30	320
`WWW-s`	16	90	12,653
`DBLP-m`	296	870	4,576
`DBLP-l`	443	1,528	7,015

Table 2: Overview of statistics of test cases: (a) average k (b) average n (c) average m.

Test Case	MDC	SC	MGP
`ACM-s`	0.84	0.82	**0.86**
`WWW-s`	**0.73**	0.59	0.65
`DBLP-m`	0.43	0.06	**0.54**
`DBLP-l`	0.4	0.05	**0.44**

Table 3: Average precision.

5.2 Evaluation Metrics. To evaluate competitive clustering methods, each cluster $c_i \in C_{i=1,..,k}$ is assigned with the most dominant label in c_i. Then, we measure the *precision* and *recall* for c_i as follows [18]:

$$Precision(c_i) = \frac{\Sigma_{i=1}^{i=k}\alpha(c_i)}{\Sigma_{i=1}^{i=k}(\alpha(c_i) + \beta(c_i))}$$

$$Recall(c_i) = \frac{\Sigma_{i=1}^{i=k}\alpha(c_i)}{\Sigma_{i=1}^{i=k}(\alpha(c_i) + \gamma(c_i))}$$

where $\alpha(c_i)$ denotes # of entities correctly assigned to c_i, $\beta(c_i)$ denotes # of entities incorrectly assigned to c_i, and $\gamma(c_i)$ denotes # of entities incorrectly not assigned to c_i.

5.3 Results. First, let us consider how accurate three methods are. Table 3 shows precisions of MDC,

Test Case	MDC	SC	MGP
`ACM-s`	0.57	**0.82**	0.64
`WWW-s`	0.36	**0.57**	0.36
`DBLP-m`	**0.35**	0.06	0.34
`DBLP-l`	**0.3**	0.05	0.24

Table 4: Average recall.

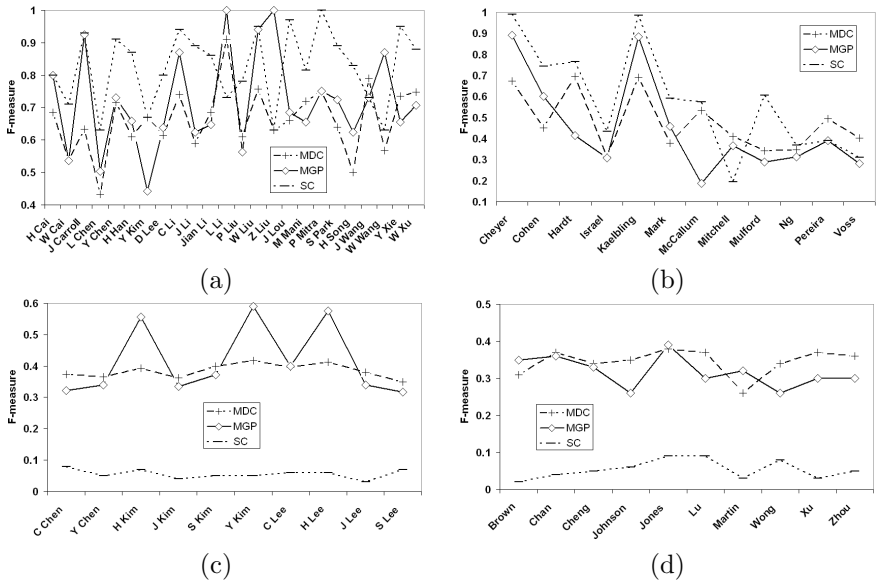

Figure 1: F-measure of (a) `ACM-s`, (b) `WWW-s`, (c) `DBLP-m`, and (d) `DBLP-l`.

SC, and MGP in four test cases. MGP shows better average precisions than both SC and MDC for three cases. On the other hand, SC is not a straightforward method as shown in Table 3. Overall, the larger the data size gets, the poorer the precision becomes. This is because the name datasets of these three methods are clustered into multi classes. Note that the precision of MGP in the `WWW-s` test case and that of MDC in the `ACM-s` test case. According to Table 3, MDC is a better name disambiguation method than MGP for web page dataset, but it becomes the opposite case for citation dataset. This indicates that using TF/IDF cosine similarity to obtain edge weights between vertices (e.g., web pages or citations) is more effective in the citation dataset. Intuitively, there is likely to be stronger relationship among an author and its variants than web page datasets. That is, a document contains a number of terms only a few of which can be considered to be important to identify variants.

Table 4 illustrates the average recall of three methods. For the small datasets, SC is the winner while for the large datasets, MDC is the winner – MGP is always the 2nd. While MGP performs coarsening and un-coarsening steps for the initial graph partitioning, the graph is approximately transformed to smaller sized one. These processes can decrease the recall of MGP in the large-scale test cases. Please note that MGP outperforms MDC in precision but MDC shows better recall than MGP. This indicates that there exists a trade-off between MGP and MDC in clustering. In conclusion, although MGP is not the clear winner for all test cases in both precision and recall, it appears to show pros of both approaches. This can be clear in Figure 1 showing F-measure (i.e., a harmonic sum of precision and recall) of four test cases.

Test Case	MDC	SC	MGP
`ACM-s`	0.12	2.2	**0.0**
`WWW-s`	2.3	4,274	**0.0**
`DBLP-m`	74	77	**0.49**
`DBLP-l`	609	169	**1.59**

Table 5: Running time (in sec.).

Table 5 shows running time of three methods for four test cases. It is clear that MGP is always winner, as we have predicted in Section 3.3. Note that both `DBLP-m` and `DBLP-l` are large-scale test cases. In `DBLP-m`, MGP is 157 times faster than SC, and in `DBLP-l` 383 times faster than MDC. Even if `WWW-s` is a small sized dataset, the running time of SC is significantly large – 4,274 sec. This is because # of dimension per vector in `WWW-s` is considerably large. For instance, the average # of dimensions a vector in `WWW-s` is about 13K while in `DBLP-l`, the largest dataset, there is on average 7K dimensions per vector. Note that, unlike MDC and MGP, k-way SC method compute k eigenvectors of a matrix. Thus, as the number of dimensions of the matrix increases, SC takes considerably more time to identify and categorize name variants. Unexpectedly, MDC is the slowest method, even worse than SC in `DBLP-l`. This is because MDC considers # of words as

its input data while SC and MGP use # of documents. Thus, the input size of MDC is significantly larger than that of SC and MGP.

6 Related Work

In literature, various researches have been proposed to investigate *the name disambiguation problem* such as [2, 12, 8] (i.e., how to cluster mixed citations into groups of distinct authors despite confusing author names). Han et al. [7] proposed two supervised learning-based approaches. Their algorithm solves the so called the *citation labeling* problem – given a citation, cross out an author name, and using the remaining citation information, recover the author name via two supervised learning methods: (1) the Naive Bayes model; (2) the Support Vector Machines. Furthermore, Han et al. [8] presented an unsupervised learning approach using K-way spectral clustering that groups different citations into different clusters, using three types of citation attributes – co-author names, paper titles, and publication venue titles. In addition, Malin [13] utilizes hierarchical clustering, relying upon the exact name similarity. However, all the approaches are not scalable to handle large-scale entities. As a scalable solution, Lee et al. [12] proposed a scalable citation labeling algorithm recently. In their algorithm, authors use the sampling-based technique to quickly determine a small number of candidates from the entire authors in a digital library.

In general, the person name disambiguation has been studied in the domain of citations. Recently Bekkerman et al. [2] proposed techniques to disambiguating collections of Web appearances using Agglomerative and Conglomerative Double Clustering.

7 Conclusion

In this paper, we have studied the name disambiguation problem and their solution. In particular, we pointed out the limitations of two state-of-the-art methods to the name disambiguation problem, and instead proposed to use multi-level graph partition techniques. Our proposal is empirically validated at the end.

References

[1] R. Bekkerman. "Multi-way Distributional Clustering (MDC)", 2005. http://www.cs.umass.edu/ ronb/mdc.html.

[2] Ron Bekkerman and Andrew McCallum. "Disambiguating Web Appearances of People in a Social Network". In *Int'l World Wide Web Conf. (WWW)*, 2005.

[3] W. Cohen, P. Ravikumar, and S. Fienberg. "A Comparison of String Distance Metrics for Name-matching tasks". In *IIWeb Workshop held in conjunction with IJCAI*, 2003.

[4] I. Dhillon, Y. Guan, and B. Kulis. "graclus software", 2005. http://www.cs.utexas.edu/users/dml/Software/graclus.html.

[5] I. S. Dhillon, Y. Guan, and B. Kulis. "A Fast Kernel-based Multilevel Algorithm for Graph Clustering". In *ACM KDD*, 2005.

[6] G. H. Golub and G. F. Van Loan. "Matrix Computations", Jun. 1996. http://www.press.jhu.edu/.

[7] H. Han, C. L. Giles, and H. Zha et al. "Two Supervised Learning Approaches for Name Disambiguation in Author Citations". In *ACM/IEEE Joint Conf. on Digital Libraries (JCDL)*, Jun. 2004.

[8] H. Han, C. L. Giles, and H. Zha et al. "Name Disambiguation in Author Citations using a K-way Spectral Clustering Method". In *ACM/IEEE Joint Conf. on Digital Libraries (JCDL)*, Jun. 2005.

[9] H. Han, H. Zha, and C. Lee Giles. "Name Disambiguation in Author Citations using a K-way Spectral Clustering Method". In *ACM/IEEE Joint Conf. on Digital Libraries (JCDL)*, Jun. 2005.

[10] B. Hendrickson and R. Leland. "An Improved Spectral Graph Partitioning Algorithm for Mapping Parallel Computations". Technical report, Tech. rep. SAND92-1460, Sandia National Lab., Albuquerque, 1992.

[11] G. Karypis and V. Kumar. "A Fast and High Quality Multilevel Scheme for Partitioning Irregular graphs". *SIAM J. Comput.*, 20(1):359–392, 1999.

[12] D. Lee, B.-W. On, J. Kang, and S. Park. "Effective and Scalable Solutions for Mixed and Split Citation Problems in Digital Libraries". In *ACM SIGMOD Workshop on Information Quality in Information Systems (IQIS)*, 2005.

[13] B. Malin. "Unsupervised Name Disambiguation via Social Network Similarity". In *SIAM SDM Workshop on Link Analysis, Counterterrorism and Security*, 2005.

[14] M. Meila and J. Shi. "A Random Walks View of Spectral Segmentation". In *Int'l Conf. on Machine Learning (ICML)*, 2001.

[15] A. Pothen, H. D. Simon, and K.-P. Liou. "Partitioning Sparse Matrices with Eigenvectors of Graphs". *SIAM J. Matrix Anal. Appl.*, 11:430–452, 1990.

[16] A. Pothen, H. D. Simon, L. Wang, and S. T. Bernard. "Toward a Fast Implementation of Spectral Nested Dissection". In *Supercomputing92*, pages 42–51, 1992.

[17] J. Shi and J. Malik. "Normalized Cuts and Image Segmentation". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[18] N. Slonim, N. Friedman, and N. Tishby. "Unsupervised Document Classification using Sequential Information Maximization". In *ACM SIGIR*, 2002.

[19] SecondString: Open source Java-based Package of Approximate String-Matching. http://secondstring.sourceforge.net/.

[20] D. Verma and M. Meila. "spectral clustering toolbox", 2003. http://www.ms.washington.edu/ spectral/.

[21] S. X. Yu and J. Shi. "Multiclass Spectral Clustering". In *Int'l Conf. on Computer Vision*, 2003.