Under consideration for publication in Knowledge and Information Systems

Scalable Clustering Methods for the Name Disambiguation Problem

Byung-Won On* [1], Ingyu Lee† and Dongwon Lee‡ [2] [3]

*Advanced Digital Sciences Center, Illinois at Singapore Pte Ltd, Singapore; †Sorrell College of Business, Troy University, USA; ‡College of Information Sciences and Technology, Pennsylvania State University, USA

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 a *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 "Masao Obama" from "Norio Obama"). 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. First, we 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 propose two scalable graph partitioning algorithms known as *multi-level graph partitioning* and *multi-level graph partitioning and merging* 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.

Keywords: Name disambiguation, Clustering methods; Mixed entity resolution; Graph partitioning; Scalability;

[1] Correspondence and offprint requests to: Byung-Won On, Advanced Digital Sciences Center, 1 Fusionopolis Way, #08-10 Connexis North Tower Singapore 138632. Email: on.byung.won@gmail.com

[2] Partially supported by NSF DUE-0817376 and DUE-0937891 awards.

[3] This paper was extended from the earlier conference paper that appeared in (On et al, 2007).

Received 06/17/2010
Revised 12/24/2010
Accepted 02/23/2011

1. Introduction

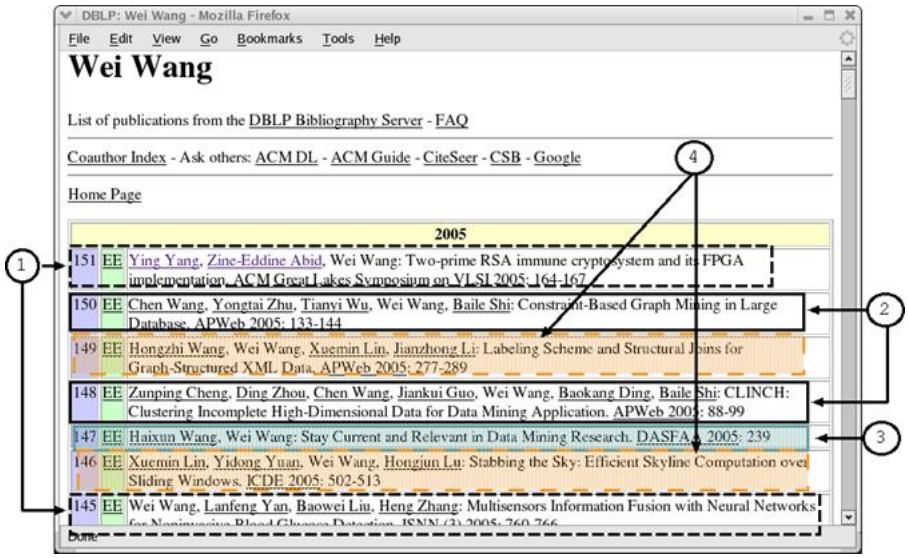
In many applications, entities need to carry a unique identifier. This identifier can be as simple as a primary key in relational database models or ISBN of books, or as complex as DNA fingerprint of people. When all applications adopt an universal identifier system such as DOI, one does not have to worry about issues related to identifiers. However, in reality, it is common to find an application that uses non-unique data values as an identifier. One of such identifiers used widely 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 name, causing 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.

Example 1. Figure 1 illustrates three real cases where entities are mixed due to their name homonyms. In Figure 1(a), citations of at least four "Wei Wang"s are mixed in DBLP where full names of scholars are used as a key. In Figure 1(b), there are at least 41 movies with the title "Love" in IMDB. Finally, Figure 1(c) shows snippets of home pages of two different "Dongwon Lee"s retrieved from Google. In a sense, full names of users are used as a key for home pages. □

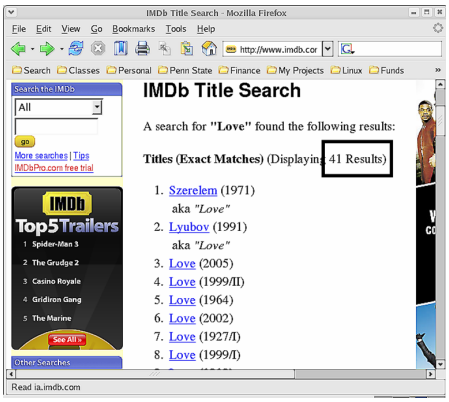
In general, one can model the name disambiguation problem as the *k-way clustering* problem. That is, given a set of mixed n entities with the same name description d, the goal of the problem is to group n entities into k clusters such that entities within each cluster belong to the same real-world group (e.g., same author or movie). For instance, in Figure 1(a), one needs to group the mixed citations (i.e., entities) of "Wei Wang"s into four groups. Similarly, in Figure 1(c), one needs to group many web pages returned from Google for the query keyword "Dongwon Lee" into two clusters – one for a faculty member at the Pennsylvania State University and the other for a graduate student at University of Minnesota.

In this paper, in particular, we study the scalability issue of the name disambiguation problem – to resolve when a large number of entities get undistinguishable due to homonyms. By and large, the scalability issue has been ignored in previous research of the 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 the limited number of feature dimensions (Bekkerman et al, 2005; Han et al, 2005). However, as data applications become more complicated and users increase rapidly, new needs arise to handle the 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 which results in high dimensional spaces in a vector space model. Therefore, it is important to have a scalable yet accurate name disambiguation algorithm.

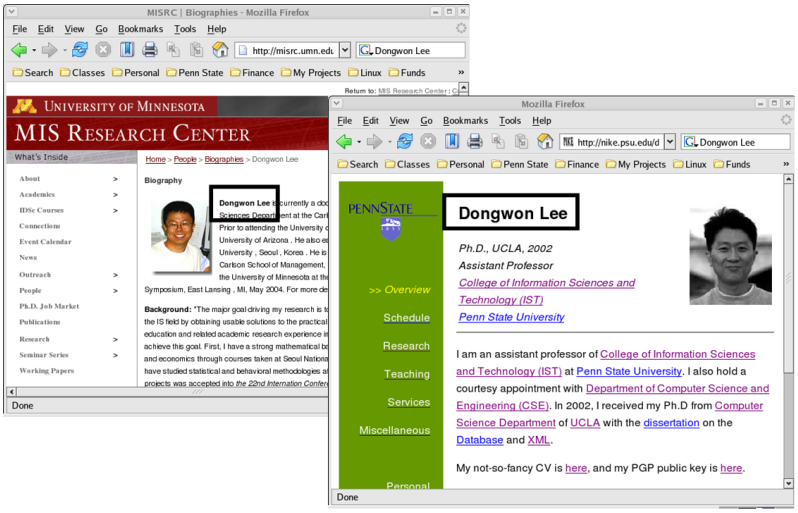
For this goal, in this paper, we carefully examine two of the state-of-the-art solutions – k-way spectral clustering (SC) (Han et al, 2005) and multi-way distributional clustering (MDC) (Bekkerman et al, 2005) – to the name disambiguation problem, and then point out their limitations with respect to their scalability. Then, we propose a scalable name disambiguation method using the **Multi-level Graph Partitioning (MGP)** algorithm to solve the large-scale



(a) Citations of at least four "Wei Wang"s are mixed in DBLP.



(b) There are at least 41 movies with the title "Love" in IMDB.



(c) Home pages of two different "Dongwon Lee"s are retrieved from Google.

Fig. 1. Examples of mixed entities due to homonyms.

name disambiguation problem. Our claim is empirically validated through our intensive empirical study in which MGP shows orders of magnitude improvement in terms of performance. For instance, according to our experimental results, MGP is about 157 times faster than SC in the `DBLP-m` data set, and is about 383 times faster than MDC in the `DBLP-l` data set. Details of the results are described in Section 5. However, the F-measure of MGP is slightly lower than both MDC and SC. These results are quite reasonable because our solution aims at efficiently clustering large-scale ambiguous name data with slightly lower clustering results, compared to competing solutions. However, we will still be able to improve the clustering results of MGP. In a sense, to maintain equivalent or more reasonable clustering results but yet to handle the scalability, we propose the **Multi-level Graph Partitioning and Merging (MGPM)** algorithm, in which the merging step is added on the multi-level graph partitioning algorithm. By approximating clusters in a scalable manner, it outperforms MDC and MGP in the F-measures. Even it has almost similar F-measures to the k-way spectral clustering (SC) which is an exact method. We will discuss the details in Section 4.

The contributions of this paper are the followings.

- We viewed a name disambiguation, which frequently occurs in digital libraries and on the web, as a hard clustering problem. To apply clustering algorithm, we categorized major clustering methods into two classes: hierarchical clustering methods and partitive clustering methods. Then, we further studied two state-of-the-art clustering methods – Multi-way distributional clustering (a hierarchical clustering variant) and k-way spectral clustering (a partitive clustering variant). In particular, we showed that such methods are not scalable when the size of ambiguous name data becomes very large.
- To address the scalability of the name disambiguation problem, we propose two multi-level algorithms. One is the multi-level graph partitioning (MGP) algorithm and the other is the multi-level graph partitioning and merging (MGPM) algorithm. In our multi-level approaches, (1) the input data is represented as an affinity graph; (2) the graph is partitioned into smaller graphs level by level; (3) the smallest graph is clustered in the end; and (4) partitioned smaller graphs are restored to the size of the original graph. According to our experimental results, MGP algorithm showed better precision, and equivalent or slightly lower recall than those of existing clustering solutions. However, it outperforms the others in terms of scalability by orders of magnitude (up to 383 times at best). Similarly, MGPM algorithm improves performance with similar or better accuracy.
- In this paper, we clearly analyzed the computational complexity of the methods as well as experimental results based on our intensive empirical study with a variety of aspects – i.e., various data sets (e.g., citation and web data sets), different sizes of data sets, and different number (and extremely skewed distribution) of cluster sizes.

The rest of this paper consists of the followings. In Section 2, we formally define name disambiguation problem. In Section 3, we investigate the multi-way distributional clustering (MDC) and k-way spectral clustering methods (SC). Section 4 describes our two multi-level algorithms: MGP and MGPM. Section 5 shows our experimental set-up and results. Section 6 discusses related work. Concluding remarks and future works follow in Section 7.

Term	Description
k	# of clusters
l	# of word tokens (delimited by spaces)
m	# of unique word tokens (i.e., m-dimensional features)
n	# of documents (i.e., entities)

Table 1. Notations.

2. Problem Definition

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

Definition 1 *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 belong to the same real-world group.* □

Clustering is the key part for name disambiguation. Consequently, we approach our problem as a **supervised** clustering problem in which the number of true clusters is known a priori in our data sets. We assume that both n and k can be a substantially large number in this problem, and the majority of input pages belongs to a single individual. We also consider our problem as a hard clustering which assigns input pages to exactly one individual cluster so that the produced clusters are not overlapped. Hard clustering can be classified as either partitioning or hierarchical approaches. Hierarchical clustering methods generate a series of nested clusters by merging simple clusters into larger ones, while partitive methods aim at finding a pre-specified number of clusters that best capture the data.

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

For our name disambiguation problem, we select two recent stat-of-the-art solutions. One is a multi-way distributional clustering method (MDC) which is one of hierarchical clustering methods and the other is a k-way spectral clustering method (SC) which is one of partitive methods. In the following, we discuss the existing methods in more depth.

3.1. Multi-way Distributional Clustering (MDC)

Bekkerman and McCallum used the *multi-way distributional clustering* method to solve the name disambiguation problem in (Bekkerman et al, 2005). Given a set of k clusters, $c_1, ..., c_k$, all word tokens of $c_{i\in[1..k]}$ are placed in a *single* cluster (a root node) while each entity of $c_{i\in[1..k]}$ is placed in a *singleton* cluster (leaf nodes). 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 process splits a cluster uniformly at random

to two sub-clusters. At the same time, the bottom-up clustering process merges each cluster with its closest neighboring cluster. The sequence of top-down and bottom-up clustering processes is pre-determined in MDC. For example, if the sequence is "Obama, Obama, Obama, Democratic, Democratic" three clustering iterations over "Obama" will be done first, followed by two iterations over "Democratic". After each iteration, each cluster are corrected based on Mutual Information – that is, it correctly clusters a random variable X (e.g., Democratic) by a joint probability distribution between X and an observed variable Y (e.g., Obama). 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).

The MDC method iteratively performs an agglomerative clustering process over terms (e.g., word tokens) and a conglomerate clustering process 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 MDC consists of two clustering approaches X and Y. X is an agglomerative clustering method over tokens such that a cluster $x \in X$ and an entity $e_i \in X_{i=1..l}$. Y is a conglomerate clustering method 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 element $e_i \in x_i$ 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

$$O(l \times n \times logn). \tag{1}$$

3.2. k-way Spectral Clustering (SC)

Han et al. used the *k-way spectral clustering* method to solve the name disambiguation problem in (Han et al, 2005). Consider a set of entities E. The spectral clustering methods consider the similarity matrix S, where $S_{i,j}$ is a similarity score between entities $e_p, e_q \in E$. As one of commonly used spectral clustering methods, Shi-Malik algorithm (Shi et al, 2000) partitions entities into two disjoint sets based on eigenvector v corresponding to the second smallest eigenvalue (i.e., Fiedler vector) of the Laplacian of S. Similarly, the Meila-Shi (Meila et al, 2001) and Han et al. (Han et al, 2005) 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 the eigenvectors.

Given a similarity matrix M, $ev_1, ...ev_k$ eigenvectors of M are computed by the k largest eigenvalues to create the matrix M' with k columns of eigenvectors of M. Finally, the rows of M' 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 $O(n^3)$ (Pothen et al, 1990)(Hendrickson et al, 1992)(Pothen et al, 1992)(Golub et al, 1996). More specifically, let us explain the time complexity of spectral clustering methods. For simplicity, we suppose a recursive spectral bisection as spectral clustering methods. At each level, we compute an eigenvalue and a corresponding eigenvector for spectral partitioning. Assume that we have n nodes in our initial graph. Then, for level i, we have 2^{i-1} subgraphs with $\frac{n}{2^{i-1}}$ node elements. If we assume the computation of eigenvector takes n^3 for n elements (Heath, 2002)(Golub et al, 1996), the total computation requires

$$O(n^3 + 2 \times \left(\frac{n}{2}\right)^3 + 4 \times \left(\frac{n}{4}\right)^3 + \cdots + n \times \left(\frac{n}{n}\right)^3) \approx O(n^3). \quad (2)$$

4. Main Proposal

In the previous section, we discussed the computational complexities of MDC and SC, and we also showed that such clustering methods are not scalable because MDC and SC have quadratic and cubic time complexities, respectively. In this section, we describe the details of MGP and MGPM algorithms so as to provide a scalable name disambiguation solution.

4.1. Graph Formation

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. More specifically, we convert the given n entities into a graph $G = (V, W)$ as follows:

- Each entity e_i is mapped to a node $v_i (\in V)$.
- Note that e_i is the i-th document in our collection. By treating e_i as a set of word terms appearing in e_i, we apply the standard vector space model to convert e_i into an m–dimensional vector (e.g., $X = (\alpha_1, ..., \alpha_m)$)[4]. If the j-th token in the entire token space appears in an entity e_i, then α_j is the TF/IDF[5] weight value of the j–th token. Otherwise, $\alpha_j = 0$.
- Finally, the edge weight between two entities e_i and e_j is computed as follows (Cohen et al, 2003).

$$TFIDF(e_i, e_j) = \sum_{t \in T_{e_i} \cap T_{e_j}} V(t, T_{e_i}) \times V(t, T_{e_j}) \quad (3)$$

[4] Let us assume that a standard stemming process has been done to filter out stop words (Howard et al, 2009).

[5] By definition, a weight is a certain value normalized in terms of importance of a word token in a document. The *Term Frequency* (TF) is a measure of the importance of the term t in a particular document. Therefore, if a term t appears in a particular document frequently, the TF weight of t will be high. On the other hand, *Inverse Document frequency* (IDF) is a measure of importance across documents in a collection. If t appears infrequently in a collection, the IDF weight will be high.

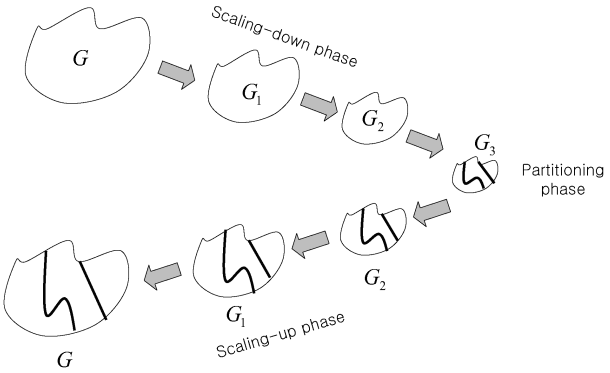


Fig. 2. The three phases of the multi-level graph partitioning algorithm.

, where t is a term (word token); T_{e_i} denotes a set of word tokens of e_i; and

$$V(t, T_{e_i}) = log(TF_{t,T_{e_j}} + 1) \times \frac{log(IDF_t)}{\sqrt{\sum_{t'}(log(TF_{t,T_{e_j}} + 1) \times log(IDF_t))}} \quad (4)$$

(symmetrical for $V(t, T_y)$). In the TFIDF similarity function, $TF_{t,T_{e_i}}$ is the frequency of t in T_{e_i}, and IDF_t is the inverse of the fraction of names in a corpus containing t.

4.2. Multi-level Graph Partitioning (MGP)

To efficiently cluster a large-scale namesake data, in our name disambiguation solution, the input data is first represented as an affinity graph G that we discussed in Section 4.1, and then the graph is approximately partitioned to subgraphs based on multi-level graph partitioning schemes, as elaborated in Figure 2. More specifically, we describe the algorithm details in Algorithm 4.1.

The multi-level graph partitioning (MGP) algorithm consists of three steps: Scaling-down, Partitioning, and Scaling-up. During the scaling-down step, the size of the graph is repeatedly decreased; in the clustering step, the smallest graph is partitioned; and during the scaling-up step, partitioning is successively refined to the larger graph. During the scaling-down 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 $O(\frac{4}{3}(20 \times k)^2((20 \times k) - 1)) \approx O((20 \times k)^3)$ (Golub et al, 1996). During the scaling-up step, the running time at each level is $O(|W|)$, where $|W|$ is # of non-zero entries (# of edges in a graph G) in the kernel matrix. Overall, the computational complexity of the MGP is

$$O(\log n + (20 \times k)^3 + |W|) \approx O(k^3 + |W|) \quad (6)$$

Note that a large graph has one million nodes and sparse edges in many practical applications. In such an application, since $k \ll n$, the computational complexity of the MGP is $O(k^3)$ which is the most efficient, compared to that of MDC and of SC.

Algorithm 4.1 *Multi-level Graph Partitioning*

`Function` MGP(G, k)

Input: An affinity graph G and # of clusters (k)

Output: A set of clusters $\{G_1, \ldots, G_k\}$

Scaling-down Phase: G is successively condensed into smaller graphs $G_1, G_2, ..., G_L$ such that $|V| > |V_1| > ... > |V_L|$, where level $i \in [1..L]$ and $|V_i|$ is the number of nodes in graph G_i. In G_i, visit each node randomly, and then merge a node v with a neighbor w that maximizes the edge weight between v and w. Once all the nodes in G_i are visited, the scaling-down process at level i is completed.

Partitioning Phase: Through the iterative scaling-down step, the smallest graph is determined such that the number of nodes is less than $20 \times k$. Then, we use a spectral algorithm (Yu et al, 2003) so as to cluster the smallest graph.

Scaling-up Phase: In order to derive partitions of the original graph G, the scaling-up step is required in which the clustered graph is repeatedly projected to a larger graph. Suppose that the size of G_i was decreased into that of G_{i+1} in the scaling-down phase. Furthermore, two nodes v and w of G_i were bound to a single node $< v, w >$ in G_{i+1}. Then, the node $< v, w >$ was partitioned into a cluster c in the partitioning phase. In the scaling-up phase, nodes 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 refining algorithm. According to (Dhillon et al, 2005), graph clustering objective functions (e.g., Ratio Association) can be transformed into weighted kernel k-means objectives, with the node weight ($\mathbb{W}$) 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, the updated clusters are computed by

$$argmin_k(K_{xx} - \frac{2 \times \Sigma_{y \in c_i^{(k)}} \times \mathbb{W}_y \times K_{xy}}{\Sigma_{y \in c_i^{(k)}} \times \mathbb{W}_y} + \frac{\Sigma_{y,z \in c_i^{(k)}} \times \mathbb{W}_y \times \mathbb{W}_z \times K_{yz}}{(\Sigma_{y \in c_i^{(k)}} \times \mathbb{W}_y)^2}) \quad (5)$$

where $c_i^{(k)}$ is the k-th cluster in the i-th iteration.

4.3. Multi-level Graph Partitioning and Merging (MGPM)

To boost up the accuracy of the MGP algorithm, we propose our multi-level graph partitioning and merging (MGPM) algorithm (as an alternative method) which is fast but more accurate graph partitioning method. The MGPM algorithm is based on a multi-level graph partitioning algorithm but also allows multi-resolutions with variant levels. At each step, the algorithm decides to go further to next level only if the partitioning step produces more inter-cluster edges than intra-cluster edges. Otherwise, the MGPM algorithm stops at the current level. In the end, the MGPM algorithm generates different levels for each branch and various resolutions for each leaf node.

After dividing graphs into smaller subgraphs, we combine the subgraphs together if two subgraphs have the biggest normalized cuts. We repeatedly merge the subgraphs until the number of subgraphs is equal to the given number of clusters (k). Algorithm 4.2 describes the details of the MGPM algorithm. After constructing a graph G, we use a spectral bisection algorithm to partition G into two subgraphs G_1 and G_2. Once partitioning G, we compute the normalized cut value between G_1 and G_2. If the normalized cut value is bigger than a pre-defined threshold value, we stop the partitioning process on that branch. Otherwise, we invoke the same process to G_1 and G_2 recursively. This partitioning step generates different levels for each branch, and the resolutions of each branch is also different. Once we have gone through the partitioning phase, we call `Function` Merge(Partition(G), k) to obtain the k clusters as outcome. The

Algorithm 4.2 *Multi-level Graph Partitioning and Merging (MGPM)*

Graph Partitioning Phase:

`Function` Partition(G)

Input: An affinity graph G

Output: A set of clusters $\{G_1, \ldots, G_i, \ldots, G_j, \ldots G_s\}$

(1) Divide G into two subgraphs G_i and G_j using spectral bisection [6].

(2) Calculate the normalized cut value between G_i and G_j, defined as

$$NormCut(G_i, G_j) = \frac{Cut(G_i, G_j)}{Edge(G_i) + Edge(G_j)} \tag{7}$$

where $Cut(G_i, G_j)$ is # of edges connecting G_i and G_j and $Edge(G_i)$ is # of edges within G_i.

(3) **if** $NormCut(G_i, G_j) \leq \Phi$
then
Partition(G_i).
Partition(G_j).
else
Stop.

Graph Merging Phase:

`Function` Merge(Partition(G), k)

Input: (1) # of clusters k and (2) a set of clusters $\{G_1, \ldots, G_i, \ldots, G_j, \ldots G_s\}$

Output: A set of clusters $\{G_1, \ldots, G_k\}$

(1) **for** $G_i \in \{G_1, \ldots, G_s\}$
Compute $Edge(G_i)$
for $G_j \in \{G_1, \ldots, G_s\}$
Compute $Edge(G_j)$ and $Cut(G_i, G_j)$.

(2) **while** $s \neq k$
if $argmax NormCut(G_i, G_j)$ such that $G_i, G_j \in \{G_1, \ldots G_s\}$ **then**
Merge G_i and G_j into $G_{i,j}$.

number of subgraphs (s) depends on the resolution we select during the partitioning phase. During the merging phase, we construct a pairwise normalized cut table as illustrated in Algorithm 4.2. Then, we merge two subgraphs which have the largest normalized cut value. We repeatedly merge subgraphs until s reaches to the given number of clusters (k).

The graph partitioning step of MGPM using the spectral bisection method is significantly fast. In most cases of practical interest, the Laplacian is a sparse matrix, in which the principal eigenvectors can be computed more rapidly using the Lanczos method. The running time for the Lanczos method to find the Fiedler vector goes approximately as $O(\frac{\varepsilon}{\lambda_3 - \lambda_2})$, where ε denotes the number of edges in the graph, and hence it can be very fast (Newman, 2004). During the merging phase, we compute a ratio between the number of edges between G_i and G_j, and the number of edges connecting G_i and G_j for each pair $i, j = 1, \ldots,$ and s where s is the number of subgraphs in the partitioning phase. It requires $\frac{1}{2}s^2$ computations until s reaches to k which is a given number of clusters. Therefore, the total computation will be $O(\frac{1}{2}(s-k) \times s^2)$, where s depends on the resolution in the partitioning phase and k is the number of clusters in the solution. Since our partition resolution s is very close to k, the execution cost of the merging phase becomes minimal.

Algorithm 4.3 *Approximating the number of clusters.*

Input: (1) Graph an affinity graph G. **Output:** # of clusters.

numClus(G**)**
  for $\delta = 0.01; \delta \leq 1; \delta = \delta + 0.01$ **do**
    Generate a subgraph G_i by removing links such that link weights $\leq \delta$
    $L(\delta) = \#$ of disconnected graph segments in G_i
  Project the L values to the coordinate system (x-axis: δ and y-axis: $L(\delta)$)
  if the L values are on *concave* **then**
    Return $L(\delta)$ such that $argmax|L(\delta) - L(\delta + 0.01)|$
  if the L values are on *convex* **then**
    Return $L(\delta + 0.01)$ such that $argmin|L(\delta) - L(\delta + 0.01)|$
  if the L values are on *linear* **then**
    Return $\frac{L(\delta)+L(\delta+0.01)}{2}$

4.4. Discussion

Our proposed approach is based on the assumption that the number of clusters is known in advance. However, this is not available in many applications. To surmount this challenging issue, in our recent paper (On et al, 2009), we proposed a novel unsupervised method of which goal is to approximately estimate the number of possible clusters a priori based on Algorithm 4.3. In this section, we summarize the key concept of our solution. For details, please refer to (On et al, 2009).

The main idea of the algorithm is to gradually disconnect several edges based on the connectivity between nodes and to analyze the patterns of segmented subgraph sequences. Assume $L(\delta)$ and $L(\delta + 0.01)$ are number of subgraphs with δ and $\delta + 0.01$, respectively. Then, we assume that the number of clusters are approximately located when the difference between $L(\delta)$ and $L(\delta + 0.01)$ is drastically changed. We observe three different types of graph segment sequences with an incremental threshold value δ: gradually increasing, gradually decreasing, and staying approximately as a constant. In the gradually increasing sequence (convex), the sequence reaches the maximum difference when the difference starts to decrease. For the gradually decreasing sequence (concave), the sequence reaches the minimum difference when the difference starts to increase. Based on this property, our algorithm assumes that the number of cluster will be close to the maximum difference in the number of subgraph sequences when it shows concave function, the minimum difference when it shows convex function, and the average of these two when it stays approximately as a constant (linear function). In this section, we also show in Table 2 that the relative error between numClus and ground truth which is defined as $|\frac{\#\text{ of clusters estimated by numClus} - \#\text{ of clusters in ground truth}}{\#\text{ of clusters in ground truth}}|$ is ten times smaller than that of between hierarchical clustering and ground truth.

5. Experimental Validation

For the MDC, SC, and MGP methods, we used the implementation of (Bekkerman et al, 2005), (Verma et al, 2003), and (Dhillon et al, 2005), respectively. For the implementation of TF/IDF cosine similarity, we used SecondString (SecondString, 2003). All experimentation were done on 4×2.6Ghz Opteron processors with 32GB of RAM.

	Hierarchical clustering	numClus
Adam Cheyer	18.5	3.5
William Cohen	4.9	0.4
Steve Hardt	6.5	0.5
David Israel	1.63	0.37
Leslie Pack Kaelbling	28.5	0
Bill Mark	6.25	0.13
Andrew McCallum	3.19	0.25
Tom Mitchell	0.78	0.27
David Mulford	3	1.38
Andrew Ng	0.23	0.19
Fernando Pereira	0.74	0.21
Lynn Voss	0.54	0
Average	6.23	0.6

Table 2. Relative error between each method and ground truth of `WWW-s`.

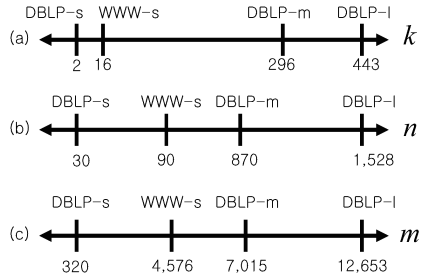


Fig. 3. `Overview of statistics of data sets: (a) average` k `(b) average` n `(c) average` m.

5.1. Experimental Set-up

For evaluation, we have used four data sets – two small and two large data sets from real examples. Figure 3 illustrates the overall statistics of the data sets and Table 3 shows the details of the data set.

– First, `DBLP-s` is a real test case that we have gathered from the DBLP 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 result. For instance, Figure 1(a) illustrates a collection of mixed citation by four "Wei Wang"s in DBLP. We collected 24 real examples as shown in Table 3, and manually checked their correctness.
– `WWW-s` is a small-sized test case using the 1,085 web pages that (Bekkerman et al, 2005) used. In 2004, the authors extracted 12 person names from Melinda Gervasio's email directory. Then, 100 top-ranked web pages ((Wu et al, 2008)) of each person name were retrieved from Google, cleaned, and manually labeled by the authors. The resulting data set consists of 1,085 web pages, 187 different persons, and 420 relevant pages. Table 3 shows the statistics of the data set. For instance, when "Tom Mitchell" is searched as a query to Google, top-92 web pages are retrieved. Among these 92, there are 37 namesakes to "Tom Mitchell". For example, among 92 web pages, "Tom Mitchell"s appear

as musicians, executive managers, an astrologist, a hacker, and a rabbi – 37 different kinds. That is, the 92 web pages, which should truly be categorized into 37 groups of web pages, are mixed since they all have the same name description of "Tom Mitchell". Like `DBLP-s`, `WWW-s` is a small but challenging test case with more number of clusters per case.

– Next, `DBLP-m` is a medium-sized citation test case generated from DBLP digital library. To generate an ambiguous name data set, we clustered author names from the entire DBLP citation data if two authors share the same first name initial and full last name. Then, we sorted the formed name clusters by the number of name variants. Finally, we obtained top-10 ambiguous names. For instance, # of "J. Lee" variants is 421 (top ranked), # of "S. Lee" variants is 391 (2nd ranked), # of "J. Kim" variants is 377 (3rd ranked) and so forth.
– Finally, `DBLP-l` is a large-scale citation test case, similar to `DBLP-m`, except that only the full last name is used in the blocking. Appendix shows the statistics of the data set.

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 (Slonim et al, 2002):

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

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

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.

Example 2. Table 4 illustrates an example of clustered documents. In "Cluster 2", since the most dominant label is *SriEng*, "SriEng" is assigned as the class label of the second cluster. $\alpha(c_2) = 3$, $\beta(c_2) = 2$, and $\gamma(c_2) = 1$. Therefore, the precision of Cluster 2 is $\frac{\alpha(c_2)}{\{\alpha(c_2)+\beta(c_2)\}} = \frac{3}{\{3+2\}} = 0.6$, and the recall of Cluster 2 is $\frac{\alpha(c_2)}{\{\alpha(c_2)+\gamma(c_2)\}} = \frac{3}{\{3+1\}} = 0.75$. □

5.3. Experimental Results

5.3.1. Scalability

Table 5 shows the 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 data set, 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,

	Name data set	k	n	m
DBLP-s	H Cai	2	5	89
	Wei Cai	2	7	120
	John M. Carroll	2	92	673
	Li Chen	2	60	718
	Yu Chen	2	46	594
	Hui Han	2	15	184
	Youngjae Kim	2	3	62
	Dongwon Lee	2	30	322
	Chen Li	2	31	343
	Jia Li	2	27	276
	Jian Li	2	21	284
	Lin Li	2	11	145
	Peng Liu	2	32	344
	Wei Liu	2	43	530
	Zhenyu Liu	2	8	139
	Jiebo Lou	2	34	311
	Murali Mani	2	11	131
	Prasenjit Mitra	2	11	115
	Sanghyun Park	2	18	201
	Hui Song	2	6	79
	James Ze Wang	2	33	310
	Wei Wang	4	143	1,264
	Yuan Xie	2	20	210
	Wei Xu	2	17	230
	Average	2	30	320
WWW-s	Adam Cheyer	2	97	12,146
	William Cohen	10	88	9,036
	Steve Hardt	6	81	14,088
	David Israel	19	92	11,739
	Leslie Pack Kaelbling	2	89	12,153
	Bill Mark	8	94	10,720
	Andrew McCallum	16	94	11,166
	Tom Mitchell	37	92	10,356
	David Mulford	13	94	16,286
	Andrew Ng	29	87	10,441
	Fernando Pereira	19	88	10,999
	Lynn Voss	26	89	22,706
	Average	16	90	12,653
DBLP-m	C. Chen	220	787	4,129
	Y. Chen	238	853	4,689
	H. Kim	290	713	3,931
	J. Kim	377	1,104	5,567
	S. Kim	302	847	4,469
	Y. Kim	240	559	3,376
	C. Lee	234	676	3,842
	H. Lee	242	557	3,509
	J. Lee	421	1,281	6,234
	S. Lee	391	1,320	6,011
	Average	296	870	4,576
DBLP-l	Brown	416	1,233	6,611
	Chan	478	1,310	6,225
	Cheng	451	1,508	6,936
	Johnson	437	1,630	7,604
	Jones	398	1,561	6,869
	Lu	471	1,581	7,071
	Martin	398	1,400	7,489
	Wong	450	1,730	7,022
	Xu	485	1,799	7,494
	Zhou	441	1,532	6,824
	Average	443	1,528	7,015

Table 3. Statistics of our four data sets.

```
Cluster 1:
           ID#1_Other.html
Cluster 2:
           ID#63_SriEng.html
           ID#85_SriEng.html
           ID#39_Player.html
           ID#1_Lawyer.html
           ID#40_SriEng.html
Cluster 3:
           ID#6_PSUProf.html
           ID#8_SriEng.html
```

Table 4. An example of clustered web documents (i.e., entities)(i) each file denotes a web page; (ii) the filename of each web page is ID#*document ID_document label.html*; and (iii) html tags and stop words are removed in each web page.

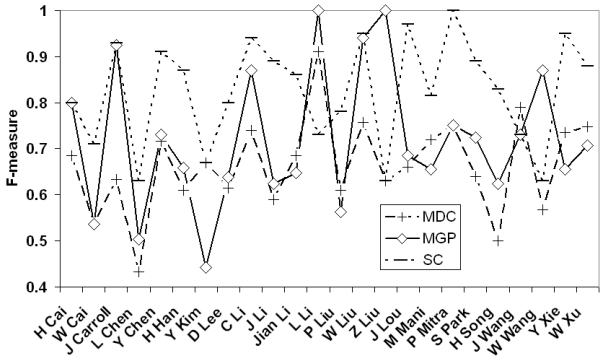
Data set	MDC	SC	MGP
DBLP-s	0.12	2.2	**0.0**
WWW-s	2.3	4,274	**0.0**
DBLP-m	74	77	**0.49**
DBLP-l	609	169	**1.59**

Table 5. Summary of running times.

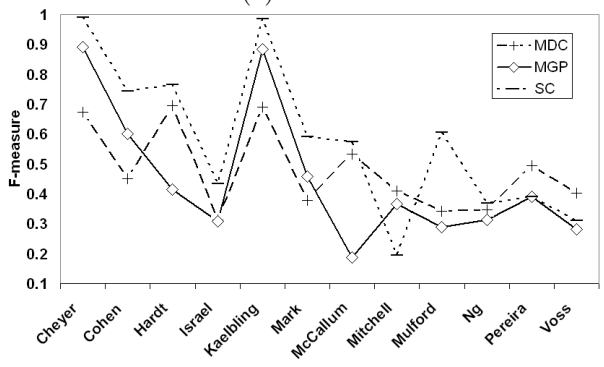
the average # of dimensions a vector in WWW-s is about 13K while in DBLP-l, the largest data set, there is on average 7K dimensions per vector. Note that, unlike MDC and MGP, k-way SC method compute k eigenvectors of a matrix. Thus, as the number of the matrix dimensions increases, SC takes considerably 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 those of SC and MGP.

5.3.2. Effectiveness of MGP

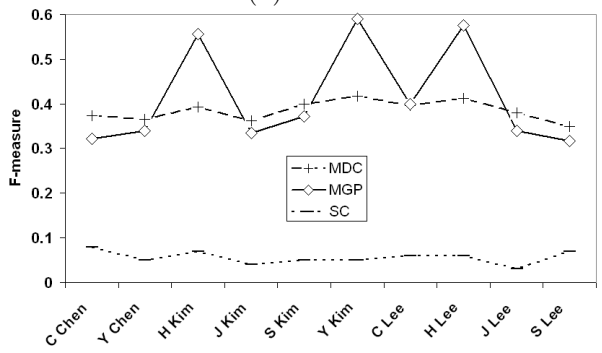
Table 6 shows the precisions of MDC, SC, and MGP in four test cases. MGP shows better average precisions than both SC and MDC in three cases. On the other hand, SC is not a straightforward method as shown in Table 6. Overall, the larger the data size gets, the poorer the precision becomes. This is because the name data sets of these three methods are clustered into multi classes. Note the precision of MGP in the WWW-s test case and that of MDC in the DBLP-s test case. According to Table 6, MDC is a better name disambiguation method than MGP for web page data sets, but it becomes the opposite case for citation data sets. This indicates that using TF/IDF cosine similarity to obtain edge weights between nodes (e.g., web pages or citations) is more effective in the citation data sets. Intuitively, there is likely to be the stronger relationships between an author and the variants than those of web page data sets. That is, a document contains a number of terms only a few of which can be considered to be important to identify variants.



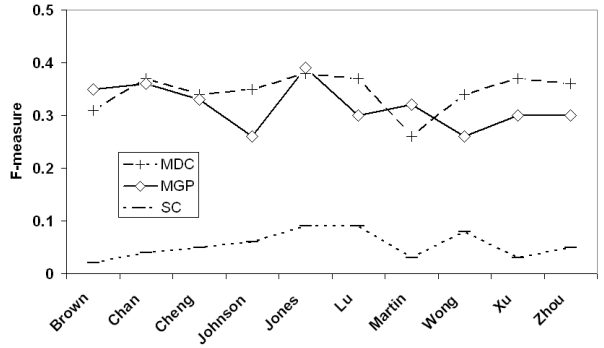
(a) DBLP-s



(b) WWW-s



(c) DBLP-m



(d) DBLP-l

Fig. 4. F-measures of our various data sets.

	Data set	MDC	SC	MGP
Average precision	DBLP-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**
Average recall	DBLP-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
Average F-measure	DBLP-s	0.68	**0.82**	0.73
	WWW-s	0.48	**0.58**	0.46
	DBLP-m	0.39	0.06	**0.42**
	DBLP-l	**0.34**	0.05	0.31

Table 6. Summary of results.

Table 6 illustrates the average recall of the three methods. For the small data sets, SC shows better performance but MDC outperforms for large data sets – MGP is always the second one. While MGP performs scaling-down and scaling-up steps for partitioning a given graph, the graph is approximately transformed to smaller sized one. These steps 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 4 showing F-measure (i.e., a harmonic mean of a precision and a recall) of small and large test cases.

Figure 5 (a)-(c) illustrate F-measure changes with the variation of n in three test cases, DBLP-s, DBLP-m, and DBLP-l, respectively. Similarly, Figure 5 (d)-(f) show F-measure changes with the variation of m in the test cases. Note that as # of citations and # of dimensions of a vector are increased, F-measures are decreased considerably. For instance, Figure 5 (b), in case of $n = 500$, MGP shows about 0.6 as the F-measure. On the other hand, when $n = 900$, F-measure is less than 0.4. Figure 5 (c) shows the similar pattern to (b). In addition, as shown in Figure 5 (d) and (e), when # of dimensions of a vector is increased, the F-measures are reduced. In particular, according to Figure 5, the SC method is the worst, compared to MDC and MGP.

5.3.3. Effectiveness of MGPM

To evaluate our proposed MGPM method, we used the WWW-s data set. Since all name data sets in DBLP-s have only two true clusters (k=2) except the "Wei Wang" name data set, all clustering methods such as MDC, SC, MGP, and MGPM are likely to cluster entities well, compared to DBLP-m, DBLP-l, and WWW-s. In contrast, since DBLP-m and DBLP-l have a number of true clusters ($k \geq$ a few hundreds), the accuracies of all the methods are poor. In Section 5, our experimental results supported this expectation. In this section, we mainly have the interest in the improvement of the accuracy of our MGPM algorithm. Thus the three DBLP data sets are not challenging to evaluate the MGPM method. Thus we focus on WWW-s for our experiments. In particular, each name data set

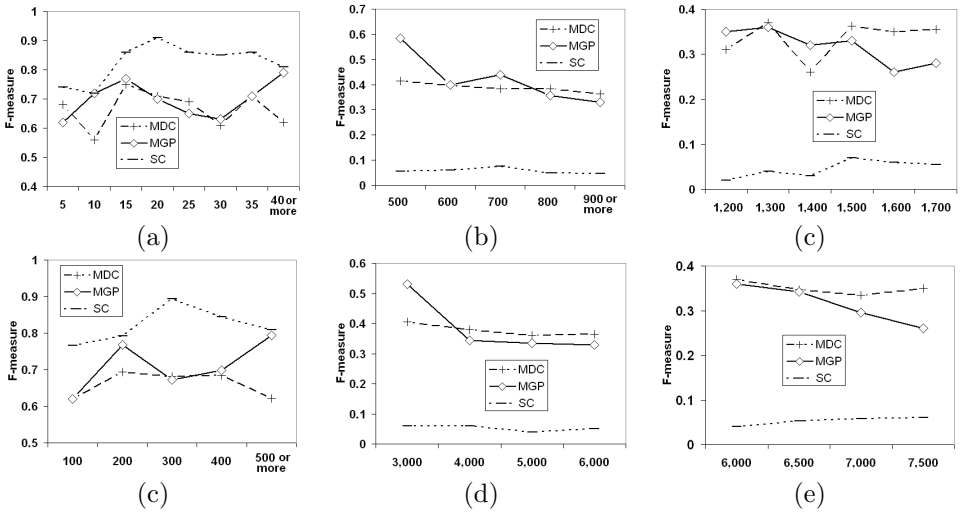


Fig. 5. (a-c) F-measure changes with the variation of n in `DBLP-s`, `DBLP-m`, and `DBLP-l`; (d-f) F-measure changes with the variation of m in `DBLP-s`, `DBLP-m`, and `DBLP-l`.

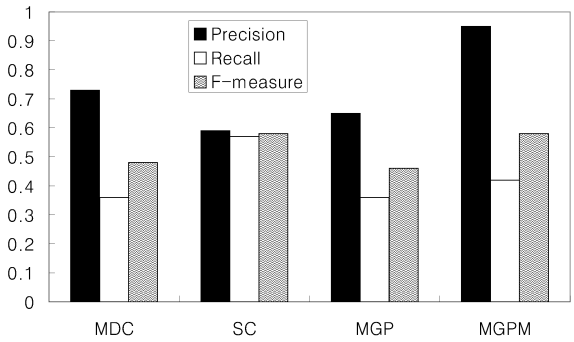


Fig. 6. Average precisions, recalls, and F-measures of the four methods.

in WWW-s has a small number of entities (but more than 2) with large contents. For instance, each web document is likely to have a number of word terms as features. In addition, the size of clusters is often skewed in each name data set. For example, the "Cohen" name data set has 10 clusters, where one cluster has 67 web documents but the other 7 clusters have only one web document in each. Due to these characteristics, WWW-s [7] is a challenging data set in order to evaluate the accuracy of the clustering methods. Thus we experimented the four

[7] According to our intensive inspection to the clustering results, rather than the small sample population of each name data set, it turned out that the clustering result becomes poor if the name data set has a large number of clusters in the ground truth and also has the skewed distribution of cluster sizes. For instance, there exist 94 web pages and 13 true clusters in the "Davaid Mulford" name data set. The largest cluster has 56 web pages, while each of the other clusters has a few web pages. As shown in Figure 4, the F-measures of all three methods in the name data set are poor.

methods such as MDC, SC, MGP, and MGPM with WWW-s and measured average precisions, recalls, and F-measures used in Section 5.2.

Figure 6 shows the average precisions, recalls, and F-measures of MDC, SC, MGP, and MGPM. Overall, average precisions are higher than average recalls. This is due to the existence of small-sized clusters in the clustering result. For instance, suppose there are two entities e_1 and e_2 with which the ground truth is $\{e_1, e_2\}$. Now suppose that a certain clustering method provides us with the clustering result: cluster c_1=$\{e_1\}$ and cluster c_2=$\{e_2\}$. In this example, the precision of c_1 and c_2 is 1 in both. On the other hand, their recalls are 0.5. In this case, small-sized clusters are likely to promote higher precisions.

In the figure, please note the results of the MGPM algorithm. MGPM shows higher recall and precision than those of the MGP algorithm. This is because of the effectiveness of the merging phase in MGPM. For example, after the partitioning step of MGPM, we suppose there are two clusters c_1=$\{e_1, e_3\}$ and c_2=$\{e_2, e_4\}$ in which the ground truth sets are $\{e_1, e_2, e_4\}$ and $\{e_3\}$. In this example, the precision of c_1 and c_2 is $\frac{1}{2} = 0.5$ and $\frac{2}{2} = 1$, and the recall of c_1 and c_2 is $\frac{1}{3} = 0.34$ and $\frac{2}{3} = 0.67$. Now suppose that the merging phase of the MGPM method combines c_1 and c_2 to the same cluster c_3. In this example, the precision and recall of c_3 are $\frac{3}{4} = 0.75$ and $\frac{3}{3} = 1$. In this case, the merging step tends to boost up the accuracy of the MGPM algorithm.

Figure 6 also shows the F-measure scores. Our proposed MGPM algorithm shows the same F-measure score as the k-way spectral clustering method which outperforms both MDC and MGP. The MGPM algorithm improves about 26% over the MGP algorithm.

6. Related Work

Our problem is associated with the Entity Resolution (ER) problem which has been known as various names – record linkage (Halbert, 2008)(Hammouda et al, 2004), citation matching (Pasula et al, 2003)(Lu et al, 2006), identity uncertainty (Pasula et al, 2003), merge/purge (Hernandez et al, 1995), object matching (Doan et al, 2003)(Aygun, 2008)(Li et al, 2005)(Wan, 2008), duplicate detection (Ye et al, 2007), group linkage (On et al, 2007)(On et al, 2006) and so on. Recently, as one of specialized ER problems, (Lee et al, 2005) introduced the mixed entity resolution problem in which instances of different entities appear together because of their homonymous names in digital libraries. For instance, there is a collection of mixed citations of two "Lin Li"s at the Pennsylvania State University and University of Nebraska in DBLP. The mixed entity resolution problem is the same as the name disambiguation problem in this paper. However, we focus more on resolving mixed entities in various applications – i.e., citation and web data sets.

To separate different groups from mixed entities, clustering methods have widely been used as solutions. So far, a variety of clustering methods such as hierarchical clustering, k-means, spectral clustering, similarity propagation (Frey et al, 2008), particle swarm optimization (Cui et al, 2003), latent dirichlet allocation, etc. have been proposed to address various clustering problems in many applications. By and large, we can categorize a majority of clustering methods into two classes. One class includes hierarchical clustering methods and the other has partitive clustering variants. For example, single-linkage, complete-linkage, average-linkage, hierarchical agglomerative and conglomerative clustering meth-

Feature	SC	MDC	MGP/MGPM
Affinity graph	Yes	No	Yes
Approximation	No	Yes	Yes
Efficiency	Slow	Moderate	Fast
Global connectivity	Yes	No	Yes
Mutual information	No	Yes	No
Parameter	k	Cut-off	k
Quality	High	Low	High
Scalability	No	No	Yes
User feedback	No	No	No

Table 7. Comparison of four clustering methods (Each entry indicates that a method is relatively higher or lower than the others.).

ods belong to the class with respect to the former. From these hierarchical clustering methods, (Bekkerman et al, 2005) proposed an agglomerative and conglomerative double clustering technique in order to disambiguate namesakes appearing on the web. In their approach, the authors used link structure of web pages and a multi-way distributional clustering method. (Banerjee et al, 2007) also proposed a multi-way clustering method in relational graphs. Different types of entities are simultaneously clustered, based not only on their intrinsic attribute values but also on the multiple relations between entities. As partitive clustering methods, there are k-means, spectral clustering variants, similarity propagation, particle swam optimization, and latent dirichlet allocation, etc. From these partitive clustering methods, to solve the name disambiguation problem, (Han et al, 2005) presented a k-way spectral clustering method which groups different citations into different clusters, using three types of citation attributes (i.e., author names, paper titles, and publication venue names).

However, all of these existing name disambiguation solutions are not scalable. On the other hand, in this paper, we focus on developing scalable name disambiguation solutions based on multi-level graph partitioning schemes such as METIS (Karypis et al, 1996) and Chaco (Hendrickson et al, 1994) which are well-known solutions for the load balancing problem in the distributed and parallel computing community. METIS aims at approximately clustering a given graph into equally sized subgraphs among which the number of edges is optimized. However, such a method often suffers from poor qualities because it divides a graph into similar-sized subgraphs (Note that all of the name data sets (e.g., `WWW-s`) are likely to have extremely skewed distribution of cluster sizes). In contrast, our proposed solutions such as MGP (see the scaling-up step based on (Dhillon et al, 2005)) and MGPM (see the graph merging step) divide a graph into various-sized subgraphs which are desired in the name disambiguation problem. In Table 7, we compare three different types of clustering methods that are considered to be name disambiguation solutions in this paper.

Nowadays, there are a few trials to develop practical systems in which search results are often mixed web pages of different entities with the same name spellings (e.g., apple companies or apple fruits). The focus of the systems is to correctly cluster web pages and to display groups of web pages. Examples of such systems are (Yippy.com, 2009) and eigenCluster (Cheng et al, 2005) based on modularity for measuring the quality of combining subgraphs to a single graph. (Hong et al, 2004) presented the construction of a prototype system to support the name disambiguation problem in a proactive way. In particular, note

that unlike our solutions, these techniques are unsupervised clustering methods because the number of true clusters is not known in advance.

7. Conclusion

In this paper, we view the name disambiguation problem as the supervised hard clustering problem in digital libraries and on the web. To address this problem, we categorized most of main clustering methods into two groups of clustering methods. One group includes various hierarchical clustering methods and partitive variants belong to the other group. For each group, we selected a state-of-the-art method to investigate whether or not such a method can indeed solve the large-scale name disambiguation problem effectively and efficiently. More specifically, we discussed the multi-way distributional clustering (a hierarchical clustering variant) and the k-spectral clustering (a partitive method based on spectral analysis) methods. According to our discussions with respect to computational complexity and experimental results with various data sets, the running time of the existing clustering solutions is prohibitively expensive. It will be deteriorated when the size of data sets is very large. To improve the efficiency of clustering methods but yet optimize clusters, we propose a multi-level graph partitioning algorithm in our preliminary work (On et al, 2007). Based on an efficient graph partitioning strategy, our early proposed clustering method showed orders of magnitude improvement in terms of performance. Furthermore, during the multi-level graph partitioning, to maintain equivalent or more reasonable clustering results, compared to competing solutions, we extend the multi-level graph partitioning algorithm to which the graph merge step is added. Our proposal is empirically validated at the end. In particular, we present the multi-level graph partitioning and merging algorithm as a main contribution in this paper.

For our future direction, we will develop a prototype system of an advanced search (named as Google namesake search) for resolving mixed entities on the web. For this system, we need unsupervised methods to estimate the number of plausible clusters in advance. In fact, scalability is considerably important in such a system because search results contain a large number of web pages retrieved from Google. To address the problem, we will plan to extend our proposed algorithms in the parallel manner. In practical systems, it is infeasible for every clustering method to correctly (or perfectly) cluster web pages at all. To obtain better clustering results, we will use the concept of feedback and investigate semi-clustering problem (induced by users' feedback) in our future framework.

References

Aygun R (2008) S2S: structural-to-syntactic matching similar documents. Knowledge and information systems 16:303-329, 2008

Banerjee A, Basu S, Merugu (2007) Multi-way clustering on relation graphs. Proceedings of the SIAM data mining, November 2007

Bekkerman R, McCallum A (2005) Disambiguating web appearances of people in a social network. Proceedings of international world wide web conference, 2005

B. Takacs and Y. Demiris (2010) Spectral clustering in multi-agent systems. Knowledge and Information Systems, 2010.

Carina Dorneles, Rodgrigo Goncalves and Ronaldo Mello (2010) Approximate data instance matching: a survey. Knowledge and Information Systems, 2010

Cheng D, Kannan R, Vempala S, Wang G (2005) A divide-and-merge methodology for clustering. ACM transactions on database systems, 2005

Cohen W, Ravikumar P, Fienberg S (2003) A Comparison of string distance metrics for name-matching tasks. Proceedings of the IIWEB workshop, 2003

Cui X, Potok T, Palathingal P (2005) Document clustering using particle swarm optimization. Proceedings of Swarm Intelligene Symposium, 2005

Dhillon I, Guan Y, Kulis B (2005) A Fast kernel-based multilevel algorithm for graph clustering. Proceedings of ACM SIGKDD conference on knowledge discovery and data mining, 2005

Doan A, Lu Y, Lee Y, Han J (2003) Profile-based object matching for information integration. IEEE Intelligent systems, September/Octobor, 2-7, 2003

Frey B, Dueck D (2007) Clustering by passing messages between data points. Science vol. 315, 2007

Golub G, Van Loan C (1996) Matrix computations. Johns Hopkins university press, Baltimore, MD, USA, 3rd edition, 1996

Halbert D (2008) Record linkage. American journal of public health 36(12):1412-1416, 2008

Han J, Kamber M, Tung A (2001) Spatial clustering methods in data mining: a survey. In geographic data mining and knowledge discovery. Taylor and Francis, 2001

Han H, Giles C, Zha H (2005) Name disambiguation in author citations using a k-way spectral clustering method. Proceedings of ACM/IEEE joint conference on digital libraries, June 2005

Hammouda K, Kamel M (2004) Document similarity using a phrase indexing graph model. Knowledge and information systems 6:710-727, 2004

Heath M (2002) Scientific computing: an introductory survey. Prentice Hall, 2002

Hendrickson B, Leland R (1992) An improved spectral graph partitioning algorithm for mapping parallel computations. Technical report, SAND92-1460, Sandia National Lab., Albuquerque, 1992

Hendrickson B, Leland R (1994) The Chaco user's guide: version 2.0. Sandia, 1994

Hernandez M, Stolfo S (1995) The merge/purge problem for large databases. ACM SIGMOD/PODS conference, 1995

Hong Y, On B, Lee D (2004) System support for name authority control problem in digital libraries: OpenDBLP approach. Proceedings of European conference on digital libraies, Bath, UK, September 2004

Howard S, Tang H, Berry M, Martin D (2009) GTP: general text parser. http://www.cs.utk.edu/∼lsi/

Karypis G, Kumar V (1996) A parallel algorithm for multilevel graph partitioning and sparse matrix ordering. Journal of parallel distributed computing 48(1): 71–95

Lee D, On B, Kang J, Park S (2005) Effective and scalable solutions for mixed and split citation problems in digital libraries. Proceedings of the ACM SIGMOD workshop on information quality in information systems, Baltimore, MD, USA, June 2005

Li Z, Ng W, Sun A (2005) Web data extraction based on structural similarity. Knowledge and information systems 8:438-461, 2005

Liping Jing, Michael K. Ng and Jushua Z. Huang Knowledge-based vector space model for text clusterng. Knowledge and information systems 25:35-55, 2010

Lu W, Milios J, Japkowicz M, Zhang Y (2006) Node similarity in the citation graph. Knowledge and information systems 11:105-129, 2006

Meila M, Shi J (2001) A random walks view of spectral segmentation. Proceedings of the international conference on machine learning, 2001

Newman M (2004) Detecting community structure in networks. European physics journal B(38): 321–330

On B, Elmacioglu E, Lee D, Kang J, Pei J (2006) Improving grouped-entity resolution using quasi-cliques. Proceedings of the IEEE international conference on data mining, 2006

On B, Koudas N, Lee D, Srivastava D (2007) Group linkage. Proceedings of the IEEE international conference on data engineering, 2007

On B, Lee D (2007) Scalable name disambiguation using multi-level graph partition. Proceedings of the SIAM international conference on data mining, 2007

On B, Lee I (2009) Google based name search: Resolving mixed entities on the Web. Proceedings of the international conference on digital information management, 2009

Pasula H, Marthi B, Milch B, Russell S, Shapitser I (2003) Identity uncertainty and citation matching. Advances in neural information processing 15, Cambridge, MIT press, 2003

Pothen A, Simon H, Liou K (1990) Partitioning sparse sparse matrices with eigenvectors of graphs. SIAM journal on matrix analysis and applications 11(3): 430–452

Pothen A, Simon H, Wang L, Bernard S (1992) Toward a fast implementation of spectral nested dissection. Proceedings of the SUPERCOM, pp 42–51, 1992
SecondString: open-source java-based package of approximate string-matching. http://secondstring.sourceforge.net/
Shi J, Malik J (2000) Normalized cuts and image segmentation. IEEE transactions on pattern analysis and machine intelligence 22(8):888–905, 2000
Slonim N, Friedman N, Tishby N (2002) Unsupervised document classification using sequential information maximization. Proceedings of the SIGIR, 2002
Verma D, Meila M (2003) Spectral clustering toolbox. http://www.ms.washington.edu/ spectral/
Wan X (2008) Beyond topical similarity: a structure similarity measure for retrieving highly similar document. Knowledge and information systems 15:55-73, 2008
Wu X, Kumar V, Quinlan J, Ghosh J, Yang Q (2008) Top 10 algorithms in data mining. Knowledge and information systems 14:1-37, 2008
Ye S, Wen J, Ma W (2007) A systematic study on parameter correlations in large-scale duplicate document detection. Knowledge and information systems 14:217-232, 2007
Yippy. http://search.yippy.com/
Yu S, Shi J (2003) Multiclass spectral clustering. Proceedings of the international conference on computer vision, 2003

Author Biographies

insert photo

Byung-Won On is a senior researcher at Advanced Digital Sciences Center (ADSC), a joint research center between University of Illinois at Urbana-Champaign and A*STAR, a Singapore Government Agency. Previously, he worked as a post-doctoral fellow at the University of British Columbia and Singapore Management University. In 2007, he earned his Ph.D. degree in the Department of Computer Science and Engineering at the Pennsylvania State University. His research interests are entity resolution and data quality, social network analysis, and query optimization that are major problems of databases, data mining and information retrieval.

insert photo

Ingyu Lee is an assistant professor at Sorrell College of Business in Troy University. He received his Ph.D. at the Department of Computer Science and Engineering in Pennsylvania State University on scientific computing algorithm and software. Main research concerns developing efficient scientific computing algorithms based on mathematical modeling using high performance computing architectures and their applications to real world problems including information retrieval, data mining, and social networking. Several problems recently working on are detecting overlapped community structure and link prediction in social network, and solving name disambiguation problem on web data using linear algebra and multi-level graph partitioning, and optimizing the application software on multicore architectures.

insert photo

Dongwon Lee received his B.E. degree in computer science from Korea University in 1993, his M.S. degree in computer science from Columbia University in 1995, and his Ph.D. degree in computer science from the University of California, Los Angeles in 2002. He is currently an associate professor in the College of Information Sciences and Technology, Pennsylvania State University. His research interests include databases, XML and Web analysis, and Web services and the Semantic Web.

Correspondence and offprint requests to: Byung-Won On, Advanced Digital Sciences Center, 1 Fusionopolis Way, #08-10 Connexis North Tower Singapore 138632. Email: on.byung.won@gmail.com