Algebraic Algorithms to Solve Name Disambiguation Problem

Ingyu Lee[1], **Byung-Won On**[2], **and Seong No Yoon**[3]
[1]Sorrell College of Business, Troy University, Troy, AL., USA
[2]Department of Computer Science, University of British Columbia, Vancouver, BC., Canada
[3]College of Business Administration, Savannah State University, Savannah, GA., USA

Abstract— *When we are looking for information about a specific person, we type the name in search engines. Then, search engines return many web pages which include the given name strings. However, the resulting web pages are mixed with related information and unrelated information. This is called a name disambiguation problem. Current search engines provide links but do not distinguish related and unrelated web pages. In this paper, we described our algorithm to solve a name disambiguation problem using two linear algebraic approaches: Singular Valued Decomposition (SVD) and Nonnegative Matrix Factorization (NMF). Experiments show that using NMF algorithm yields the best performance in terms of precision, recall and Fmeasure, and applying SVD requires only 10% computation time compared to traditional K-means algorithm. Our solution with search engines will provide more precise search results than traditional search engine services.*

Keywords: Data mining, Text mining, Clustering, SVD, NMF

1. Introduction

Internet allows people to work with or to get information about someone whom we never met before in real life. Search engines are used to find information about persons we are interested in. When we are looking for information about a specific person, we type the name string in search engines. Then, search engines return many web pages including the name strings. However, some web pages include the name string by coincidence but do not contain related information. Current search engines do not distinguish between web pages described in the latter. For example, if we are looking for *Tom Mitchell* from Carnegie Mellon University, then we type the name in Google search engine. Google search engine returns web pages including *Tom Mitchell*. There are 37 different *Tom Mitchell* within the first 100 returning web pages [1]. Only *57* documents hold information about *Tom Mitchell* from Carnegie Mellon University and all the other documents hold information for different *Tom Mitchell*. We call this a name disambiguation problem.

One approach to solve the problem is providing web pages as groups of related information. Since search engines return many related and unrelated web pages, users might be mislead by bunch of unrelated web pages. To prevent this, search engines cluster related web pages and provide clustered information to users. The latter will reduce chances that leading users to unrelated web pages. Therefore, clustering algorithms such as K-means, hierarchical clustering and graph partitioning have been used to solve a name disambiguation problem [2], [3], [4], [5].

At the same time, linear algebraic approaches have been used in information retrieval applications [6], [7]. Especially, Singular Value Decomposition and Nonnegative Matrix Factorization (NMF) are popularly used in text and data mining applications. When dimensions become large, applying SVD and NMF could significantly reduce dimension with a slight loss of original information. In this paper, we applied these two algorithms to solve a name disambiguation problem and compared performance with a regular K-means algorithm. Experiments show that SVD reduces clustering time and NMF improves performance. Our solution with search engines provide more precise search results than traditional search engine services with a slight overhead.

The rest of the paper consists of the followings. Linear algebra algorithmic backgrounds are described in Section 2. Characteristics of a name disambiguation problem and metrics we are using to measure performance are described in Section 3. Experimental results with sample data name set using Singular Value Decomposition (SVD) and Nonnegative Matrix Factorization (NMF) are described in Section 4. Related researches are described in Section 5. Re-ranking clustered results are described in Section 6. Concluding remarks and future plans are followed in Section 7.

2. Linear Algebraic Algorithms

Vector Space Model (VSM) has been used in information retrieval applications to represent text documents [6], [7]. Assume we have n documents corpus and we want to represent each document using m terminologies. Then, we make n textual documents into n document vectors $d_1, d_2, \ldots, d_n$ where each document vector has m terminologies. Therefore, the term-by-document matrix A is represented as

$$A_{m\times n} = [d_1|d_2|\cdots|d_n] \qquad (1)$$

where columns are document vectors and rows are terminologies as shown in Figure 1.

To retrieve relevant information from this term-document matrix, we create query vector q and find the document d which is the closest to a query q. Euclidean distance or cosine similarity are used to measure distances between a

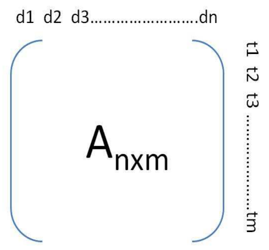

Fig. 1: Term-Document Matrix

query q and a document d defined as

$$Eucliean\_Distance(d,q) = \sqrt{\sum_{k=1,\dots,m} (d_k - q_k)^2} \quad (2)$$

and

$$\cos\theta = \frac{q^T d}{||q||_2 ||d||_2} \quad (3)$$

, respectively.

2.1 K-means Clustering

Assume we have a $m \times n$ term-document matrix A whose m rows represent terminologies and n columns represent documents as defined in the previous section. We wish to cluster n documents into k disjoint clusters $C_1, \dots, C_k$. Then, the objective function of K-means algorithm is defined as

$$\min \sum_{i=1,\dots,k} \sum_{d_j \in C_i} (d_j - m_i)^2 \quad (4)$$

where d_j is a document column set from A and m_i is a centroid of cluster C_i for $i = 1, \dots, k$.

K-means algorithm works the following way.

1) Choose random k cluster centers (centroid) $m_j, j = 1, \dots, k$ from column set of document vectors.
2) Assign each document column to the closest clusters $C_j, j = 1, \dots, k$ using Euclidean distance or cosine similarity as defined in the previous section.
3) Compute new centroid, $m_j, j = 1, \dots, k$ which are means of all points in C_j defined as

$$m_j = \frac{1}{|C_j|} \sum_{l=1,\dots,|C_j|} d_l \quad (5)$$

4) Repeat until changes in clusters C happen.

K-means algorithm is the most popular to cluster data sets because of its simplicity. In addition, the quality of K-means algorithm is known as good for general clustering problems. However, the convergence of the algorithm depends on initial centroid selection. If initial centroid is located far from the true solution set, then K-means requires lots of iteration to converge. Details of K-means and variants algorithms are found in [8], [9].

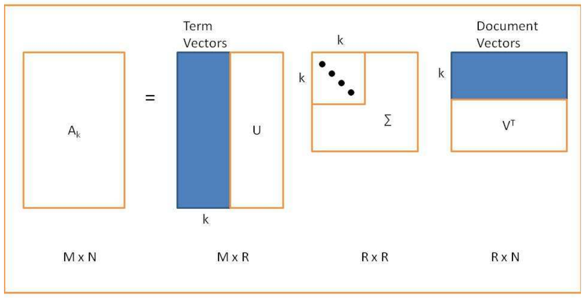

Fig. 2: Singular Value Decomposition

2.2 Singular Value Decomposition

Singular value decomposition (SVD) has been used to solve linear least squares problems, matrix rank estimation, and correlation analysis [6], [10], [11]. Given a $m \times n$ matrix A, where $m \geq n$, the singular value decomposition of matrix A is defined as

$$A = U \Sigma V^T \quad (6)$$

where U and V are orthogonal matrices which satisfy $U^T U = V^T V = I$, and Σ is a diagonal matrix whose diagonal elements are $diag(\sigma_1, \dots, \sigma_n)$.

If we have a term-document matrix A whose columns represent documents holding the terminology on each row, then matrix A is generally large and sparse in real applications. Applying K-means on this large and sparse matrix A requires lots of time and memory space. To overcome the latter, SVD has been used to represent documents with reduced number of terminologies which best represents the original matrix A.

Dimension reduced matrix A_k is noted as

$$A_k = U_k \Sigma_k V_k^T \quad (7)$$

where U_k and V_k are orthogonal matrices on reduced dimension, and Σ_k is a diagonal matrix with k singular values. In the reduced domain, V_k represents the projection of the documents into k terminology spaces. Figure 2 shows the outline result of SVD factorization. Matrix A is factorized into three matrices. Columns of the leftmost matrix U are basis vectors for terminology and columns of the rightmost matrix V represent documents with a combination of basis vectors. The diagonal matrix value represents scaling factors of each basis vector. Matrix V is a projection of our matrix A in terms of terminologies we selected as shown in Figure 2.

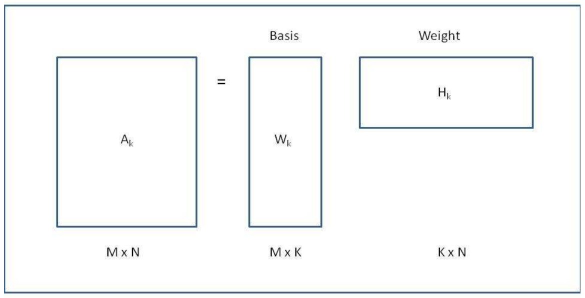

Fig. 3: Nonnegative Matrix Factorization

Using A_k in place of A improves performance by reducing dimension. A_k considers only essential components of term-by-document matrices. It also filters out the noise, and uses best rank-k approximation. The drawbacks are U_k and V_k are completely dense matrices which requires $(m+n)k$ memory space. Furthermore, truncation point k is hard to determine, and interpretation of basis vectors u_i and v_i is difficult due to mixed signs. Details of algorithms and characteristics are described in [6], [10], [11].

2.3 Nonnegative Matrix Factorization

Nonnegative Matrix Factorization (NMF) has been used in data mining applications [12], [13] since negative values in SVD matrices are hard to understand the meaning. Unlike SVD factorization, Non-negative Matrix Factorizations (NMF) uses low-rank approximation with nonnegative factors. The following shows the outline of NMF algorithm.

$$A_k = W_k H_k \tag{8}$$

where columns of W are the underlying basis vectors (i.e. each n column of A can be built from k columns of W) and columns of H give the weights associated with each basis vector such as

$$A_k e_1 = W_k H_{*1} = [w_1]\, h_{11} + [w_2]\, h_{21} + \cdots + [w_k]\, h_{k1} \tag{9}$$

where w_i is a basis and h_{ji} is a contribution weight for each basis $w_j, j = 1, \cdots, k$. The outline of the algorithm is shown in Figure 3.

Since w_is are not orthogonal, basis vectors could overlap topics. Unlike SVD, W and H can also be restricted as sparse matrices. The value w_{ij} means the contribution of term j in basis vector w_i. The value h_{i1} shows how much doc_1 is pointing in the direction of topic vector w_i. Since W and H are not unique, there are several different algorithms to factorize into W and H.

Lee and Seung applied NMF algorithm to image processing applications in [14], [15]. Berry used NMF algorithms with sparse constraints on matrix H in his paper [12], [13], and Paatero and Tapper proposed an alternative least squares NMF algorithm which is known as faster and simpler in their paper [16]. NMF has great interpretability, and comparable performance to SVD. Furthermore, sparsity constraints during factorization also allow significant storage savings. It is also scalable and possibly faster than SVD in parallel environments. However, the factorization is not unique and depends on algorithms and parameters. When a new dimension is added, it is unable to reduce the size of the basis without recomputing NMF. Figure 4 shows an alternative least squares NMF algorithm we used in this paper. In the algorithm, λ is a sparse constraints factor for matrix H.

```
function [W, H] = NMF(A)
% initialize nonnegative random matrix, W
W = abs(randn(m,k));
% initialize nonnegative random matrix, H
H = abs(randn(k,n));
for i=1 to maxiter do
    % Solve for H
    $H = (W' * W + eye(k). * \lambda)\ (W^T * A);$
    % Keep positive only
    $H = H. * (H >= 0);$
    % Solve for W
    $W^T = (H * H^T + eye(k). * \lambda)\ (H * A^T);$
    % Keep positive only
    $W = W. * (W >= 0);$
end for
```

Fig. 4: Nonnegative Matrix Factorization

3. Problem Settings and Metrics

For the evaluation of our algorithms, we used name data set from Bekkerman [1]. It has twelve personal names and each name set has different number of categories. Table 1 and Figure 5 shows the statistics of our dataset. Table 1 shows that each name set has variants number of categories. For *Adam Cheyer* and *Leslie Pack Kaebling* has *2* categories but *Tom Mitchell* has 37 different categories. In Figure 5, each color represent different clusters and sizes of bar show the number of documents belong to each cluster. The naming data set is extremely skewed in terms of cluster size. For *Adam Cheyer* and *Leslie Park Kaebling*, all documents except one document belong to the same class. Others such as *Andrew Ng* and *Voss*, one or two classes dominate the whole data set. This skewed characteristic makes it difficult to use traditional clustering algorithm.

The first step for experiments is building *term-document* matrix A from the name data set. We used TMG [17] software to generate the *term-document* matrix with normalization and stemming options. We also removed common words from the term lists. We used *term frequency* for

Name	Pages	Categories
Adam Cheyer	97	2
William Cohen	88	10
Steve Hardt	81	6
David Israel	92	19
Leslie Pack Kaebling	89	2
Bill Mark	94	8
Andrew McCallum	94	16
Tom Mitchell	92	37
David Mulford	94	13
Andrew Ng	87	31
Fernando Pereira	88	19
Lynn Voss	89	26
Total	**1085**	**187**

Table 1: Dataset Statistics. Each name has different number of documents and different number of categories.

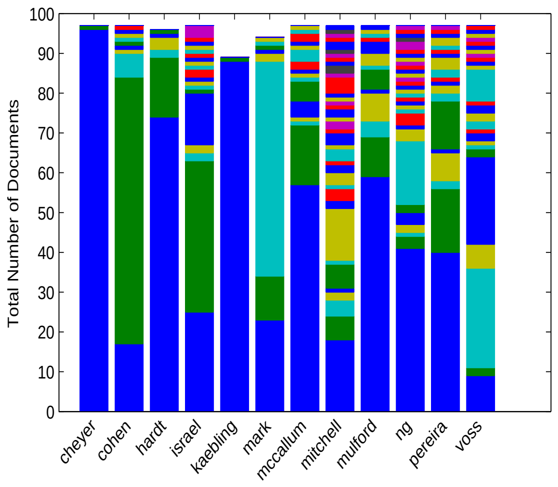

Fig. 5: Statistics of Problem Set: Name sets are extremely skewed and one class dominates the whole document set.

local term weighting and *inverse document frequency* for global term weighting, respectively. Each nonzero elements in $A(i,j)$ is defined as

$$A(i,j) = TF_{i,j} \times IDF_{i,j} \tag{10}$$

where TF is *term frequency* and IDF is *inverse document frequency*. We also normalized each column of A. After built *term-document* matrix A, we used Singular Value Decomposition (SVD) and Nonnegative Matrix Factorization (NMF) to reduce dimensions and project web documents on the reduced terminology space.

The first metric we considered is the accuracy of clustering algorithm. We defined the precision, recall and Fmeasure as in equations

$$Precision_i = \frac{CorrectlyPredict_i}{Predict_i} \tag{11}$$

$$Recall_i = \frac{CorrectlyPredict_i}{Cluster_i} \tag{12}$$

$$Fmeasure = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{13}$$

where $Predict_i$ is the results of clustering algorithm and $Cluster_i$ is a manually generated solution set. Precision is defined as the number of correctly clustered documents divided by the number of documents in the cluster. Recall is defined as the number of correctly clustered documents divided by the number documents in manually generated solution set. Fmeasure is an arithmetic mean of precision and recall.

Since the original name set is extremely skewed, the precision and recall could be high even quality of clustering results is not good. For example, *Cheyer*, one class has *95* documents out of *96* documents and the second class has only *1* document. The precision is *99%* no matter which clustering algorithms are used. To distinguish performance between clustering algorithms, we also applied RAND index which has been used in statistics community to compare two clustering results [18]. Given a set of n documents $D = \{d_1, \ldots, d_n\}$, we wish to compare two clusters: C and S. The resulting cluster of our algorithm is defined as $C = \{C_1, \ldots, C_k\}$ and manually derived solution set is defined as cluster $S = \{S_1, \ldots, S_k\}$. Then, RAND index is defined as

$$RAND = \frac{a+b}{a+b+c+d} \tag{14}$$

where a is the number of pairs of elements in D that are in the same set in C and the same set in S, b is the number of pairs of elements in D that are in different sets in C and in different sets in S, c is the number of pairs of elements in D that are in the same set in C and in different sets in S, and d is the number of pairs of elements in D that are in different sets in C and in different sets in S. Intuitively, $a+b$ is the number of agreements between C and S, and $c+d$ is the number of disagreements between C and S. RAND index has a value between 0 and 1 with 0 indicating that two data clusters do not agree on any pair of points and 1 indicating that the data clusters are exactly the same.

4. Experimental results

K-means algorithm use centroids and medois. Centroid is a computed center as defined in the previous chapter and medoid is a member of cluster which is closest to the computed centroid. Using medoids is known as more efficient than using centroids [19]. Figure 6 shows the performance of two methods. We averaged results of *100* experiments for each name data set. The x-axis shows different metrics

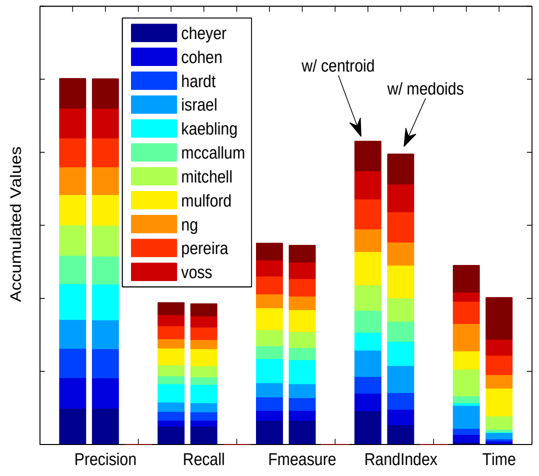

Fig. 6: Performance of two different approach: centroids vs. medoids.

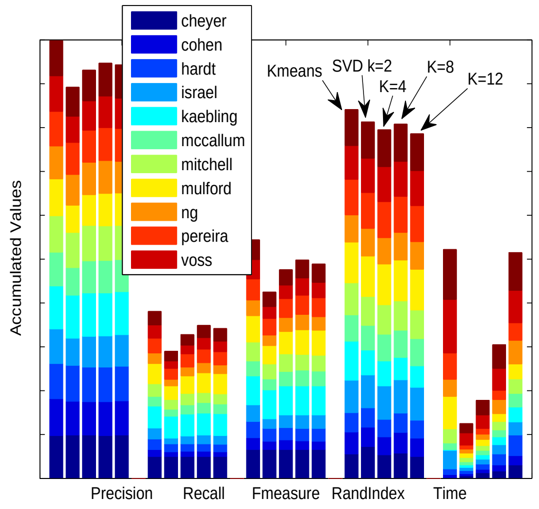

Fig. 7: Performance results of K-means after dimension reduction.

and the y-axis are accumulated value of all name sets. Two methods show similar performance in Fmeasure but using centroids shows a little bit better performance in Rand index. However, using medoids converges faster than using centroids. In this paper, we used medoids in our K-means clustering algorithm rather than using centroids since it shows similar Fmeasure with less computation time.

Figure 7 shows the performance comparison of K-means algorithm and K-means on reduced matrix when we use *k=2,4,8,12*. We experimented 100 times and computed averages for each name data set. We achieved *90%* of performance in terms of Fmeasure and Rand index with

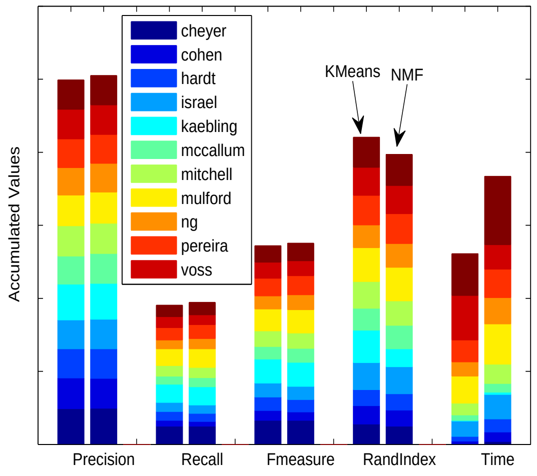

Fig. 8: Performance results of K-means and NMF.

10% execution time when we use *k=2*. Bigger k values improve performance slightly better but require more time to converge. The experimental results show that only several principal terminologies are enough to define documents in name data sets.

To measure NMF clustering algorithm, we used an alternative updating algorithm as shown in Figure 4 with λ value 0.02. The factorization iteration stops when the difference between the previous norm of H and the current norm of H is smaller than 0.01. After factorizing the matrix A into WH, we searched the biggest component from column vector H_i. Then, we assign the document with the biggest component cluster. Figure 8 shows the results of comparing performance of two methods. Using NMF shows slightly better performance in terms of precision, recall and Fmeasure but K-means shows better performance in Rand index and time. Considering the size of testing data set is small and using sparse matrix format for term-document matrix, K-means shows faster performance is not surprising.

To compare SVD and NMF with the same k dimension number, we applied K-means algorithm after reducing dimension to $m \times k$ with SVD and compared the results with NMF algorithm. The results are shown in Figure 9. NMF shows slightly better performance in precision, recall, Fmeasure and Rand index. It also much faster than SVD to converge. In addition, considering parallelizing SVD is much harder than NMF, our experiments show that NMF is a better choice if we are using the same k value. If data sets are large in parallel environment, NMF is the better algorithm but SVD with lower k gives benefits in execution time.

Experiments show that extremely skewed name data sets are hard to cluster using regular clustering algorithm. However, SVD could help in reducing dimensions with a

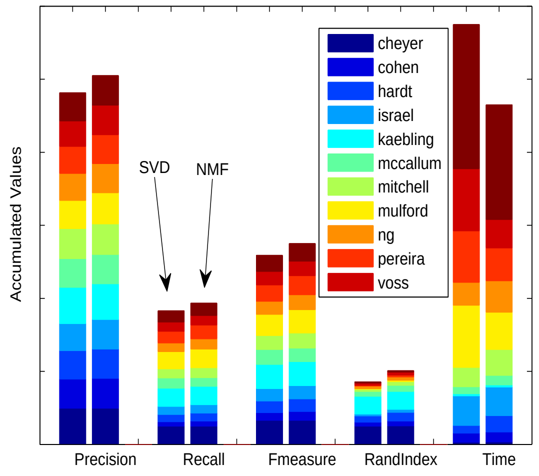

Fig. 9: Performance results SVD with K-means and NMF.

little loss of quality and NMF shows the best performance. Therefore, for time limited applications, applying SVD is helpful but NMF algorithm generates a better performance in general. Especially, considering parallelizing NMF is relatively easier than SVD, NMF is the algorithm in parallel environment.

5. Related Work

Bekkerman et. al. proposed two algorithms to disambiguate web appearances of people in a social network in their paper [3]. One is based on link structure of web pages, another using multi-way distributional clustering method. Their algorithms show more than *20%* improvement in Fmeasure. Minkov et. al. used lazy graph walk algorithm to disambiguate names in email documents in their paper [4]. They provided a framework for email data, where content, social networks and a timeline to integrated in a structured graph. Banerjee et. al. proposed multi-way clustering on relation graphs in [2]. Different types of entities are simultaneously clustered based not only on their intrinsic attribute values, but also on multiple relations between entities. On and Lee used multi-level graph partitioning methods to provide a scalable name disambiguation solution in their paper [5].

In authors awareness, this is the first paper to provide an algebraic approach in a solving name disambiguation problem. In addition, we analyzed the name disambiguation problem characteristics and used Rand Index to measure the performance of extremely skewed clustering problem. Our experiment results show some promising results in choosing a proper algorithm based on the problem and computational environments.

6. Discussion

As the results of clustering, relevant web pages are clustered to the same group. For instance, there are four different people with the same name spellings, shown in Table 2. In the example, the total number of web pages related to "Tom Mitchell" is *13*, where four web pages (i.e., A, B, C, and D), three ones (i.e., E, F, and G), five ones (i.e., H, I, J, X, and Y), and one web page are associated with professor at CMU, reporter at CNN, musician at Shady record company, and minister at Kansas city, respectively. Suppose that web pages are ranked by PageRank scores like the third column in Table 2. The current search result of Google is A, E, J, F, G, I, H, D, Y, B, X, C, and Z. These *13* web pages are clustered to four groups in terms of our clustering scheme, and furthermore we need to re-arrange the four groups in a certain order (like PageRank). Let us denote this process as *re-ranking* of clusters.

This is a considerably challenging issue in this paper. For instance, note the four web pages of "Tom Mitchell" at CMU. Web page A is firstly ranked by Google PageRank algorithm, while the rest pages are mostly located in bottom – B (10^{th}), C (12^{th}), and D (8^{th}). In this case, it is hard to determine which position the cluster (labeled as CMU/Prof) should be rank among the four clusters (labeled as CMU/Prof, CNN/Rep, Shaddy/Mus, and Kansas/Min). To address this problem, we propose an approach based on the hypothesis that *re-using ranking information generated by Google is sufficiently effective*. For this, we consider four different methods as follows:

- Consider *relative ranking positions* of web pages per cluster which is defined as

$$ClusterRank_j = \frac{\sum_{i \in Cluster_j} PageRank_i / N_{doc}}{N_{cluster_j}} \tag{15}$$

where $Cluster_j$ is the number of pages in cluster j and N_{doc} is the total number of pages. For example, take a look at the cluster, named as "CMU/Prof". We can compute the cluster ranking by

$$\frac{\frac{PageRank(A)}{13} + \frac{PageRank(B)}{13} + \frac{PageRank(C)}{13} + \frac{PageRank(D)}{13}}{4}$$
$$= \frac{\frac{1}{13} + \frac{10}{13} + \frac{12}{13} + \frac{8}{13}}{4}$$
$$= \frac{(0.08+0.77+0.92+0.62)}{4} = \frac{2.39}{4} = 0.6$$

That is, $ClusterRank(CMU/Prof) = 0.6$. Finally, the clusters are re-ordered by the $ClusterRank()$ scores in the ascending order.

- Consider the *highest ranking position* of web pages per cluster which is defined as

$$ClusterRank_j = \min_{i \in Cluster_j} PageRank_i. \tag{16}$$

For instance, the cluster of "CMU/Prof" is ranked in the first position due to the ranking position of A. In other words, $ClusterRank(CMU/Prof) = 1$. Finally, the

Cluster Label	Web Page ID	PageRank
Tom Mitchell Professor CMU	A	1
	B	10
	C	12
	D	8
Tom Mitchell Reporter CNN	E	2
	F	4
	G	5
Tom Mitchell Musician Shady Records	H	7
	I	6
	J	3
	X	11
	Y	9
Tom Mitchell Minister Kansas City	Z	13

Table 2: An example of re-ranking in our context.

clusters are re-ordered by the $ClusterRank()$ scores in the ascending order.

- Consider the *Top-k ranking positions* of web pages per cluster which is defined as

$$ClusterRank_j = \frac{\sum_{TopK \in Cluster_j} PageRank_{topk}/N_{doc}}{K} \quad (17)$$

where $TopK$ is the highest k rank values in cluster j. For instance, the cluster of "CMU/Prof" has top-2, $PageRank(A)$ and $PageRank(D)$. Then,

$$\begin{aligned} ClusterRank_j &= \frac{\frac{PageRank(A)}{13} + \frac{PageRank(D)}{13}}{2} \\ &= \frac{(0.08 + 0.62)}{2} = 0.35. \end{aligned}$$

Finally, the clusters are re-ordered by the $ClusterRank()$ scores in the ascending order.

- Consider the *median ranking position* of web pages per cluster which is defined as

$$ClusterRank_j = median\{Cluster_j\} \quad (18)$$

As an example, $ClusterRank(Shaby/Mus) = 7$ due to $PageRank(J) = 3$, $PageRank(I) = 6$, $PageRank(H) = 7$, $PageRank(Y) = 9$, and $PageRank(X) = 10$. Finally, the clusters are re-ordered by the $ClusterRank()$ scores in the ascending order.

7. Concluding Remarks

In this paper, we characterized and analyzed a name disambiguation problem. We also applied algebraic approaches to solve a name disambiguation problem using SVD and NMF algorithms. Using SVD showed drastic improvement in terms of execution time. It requires only *10%* of execution time if we are using smaller number of *k*. Using NMF shows a little bit better performance in terms of precision, recall, Fmeasure and Rand Index. Considering that NMF is relatively easier to parallelize than other algorithms, NMF is an algorithm to use in parallel environment with large name data sets.

Current implementation is based on a single machine and it could not handle huge number of data sets. In the future, we are considering to implement our algorithms, especially NMF algorithm, on a clustered system to handle large number of data sets. In addition, several variation metrics are needed to measure performance of extremely skewed clustering problem. We are trying to provide better quality metrics for a name disambiguation problem. We also proposed several re-ranking algorithms of clustered results. However, we are still working on the details of several re-ranking algorithms.

References

[1] R. Bekkerman, "Name data set," http://www.cs.umass.edu/ ronb.

[2] A. Banerjee, S. Basu, and S. Merugu, "Multi-way clustering on relation graphs," in *Proceedings of SIAM Data Mining 2007*, 2007.

[3] R. Bekkerman and A. McCallum, "Disambiguating web appearances of people in a social network," in *Proceedings of International World Wide Web Conference Committee*, 2005.

[4] E. Monkov, W. Cohen, and A. Y. Ng., "Contextual search and name disambiguation in email using graphs," in *Proceedings of SIGIR*, 2006.

[5] B. On and D. Lee, "Scalable name disambiguation using multi-level graph partition," in *Proceedings of SIAM Data Minings*, 2006.

[6] M. Berry and M. Browne, *Understanding Search Engines*. SIAM, 2005.

[7] M. Berry, S. Dumais, and G. OBrien, "Using linear algebra for intelligent information retrieval," *SIAM Review*, vol. 37, pp. 573 – 595, 1995.

[8] J. Han, M. Kamber, and A. Tung, *Spatial clustering methods in data mining: A survey. In Geographic Data Mining and Knowledge Discovery*. Taylor and Francis, 2001.

[9] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, 1967.

[10] G. Golub and C. V. Loan, *Matrix Computations*, 3rd ed. Baltimore, MD: Johns Hopkins University Press, 1996.

[11] M. Heath, *Scientific Computing: An Introductory Survey*. New York: McGraw Hill, 2002.

[12] M. Berry, M. Browne, A. Langville, V. Pauca, and R. Plemmons, "Algorithms and applications for approximate nonnegative matrix factorization," *Computational Statistics and Data Analysis*, vol. 52, pp. 155 – 173, 2007.

[13] F. Shahnaz, M. Berry, V. Pauca, and R. Plemmons, "Document clustering using nonnegative matrix factorization," *Information Processing and Management: an International Journal*, vol. 42, pp. 373–386, 2006.

[14] D. Lee and H. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, pp. 788 – 791, 1999.

[15] ——, "Algorithms for non-negative matrix factorization," in *NIPS*, 2000.

[16] P. Paatero and U. Tapper, "A non-negative factor model with optimal utilization of error estimates of data values." *Environmetrics*, vol. 5, pp. 111–126, 2006.

[17] D. Zeimpekis and E. Gallopoulos, *TMG: A MATLAB toolbox for generating term document matrices from text collections*, 2006.

[18] W. M. Rand, "Objective criteria for the evaluation of clustering methods," *Journal of the American Statistical Association*, vol. 66, pp. 846–850, 1971.

[19] L. Kaufman and P. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley, 1990.