Framework For Mixed Entity Resolving System Using Unsupervised Clustering

Byung-Won On[1], Ingyu Lee[2]
[1]Singapore Management University
Singapore

[2]Troy University
AL. USA

Journal of Digital Information Management

ABSTRACT: *During web search, confusion can happen due to homonym when users use non-unique values as a search term of an entity. Especially, when parts of names of an entity were used as its identifier, we call a* ***mixed entity resolution*** *problem whose goal is to clear out the erroneous entities. For example, if only last name is used as an identifier, we cannot distinguish "Vannessa Bush" from "George Bush." Mixed entity resolution problem is common among Web pages data. In this paper, to resolve aforementioned mixed entities on the Web, we propose a prototypical system which includes a web service based interface, unsupervised clustering scheme, and cluster ranking algorithms. In particular, since the correct number of clusters is often unknown, we study a state-of-the-art unsupervised clustering solution based on propagation of pairwise similarities of entities. Experimental results show that our approach outperforms main competing solution.*

Categories and Subject Descriptors
H.3.3 [Information Search and Retrieval]; Clustering: **H.5.3 [Group and Organization Interfaces**] ; Web-based interaction

General Terms: Data mining, Web search, web services, Web page clustering

Keywords: Mixed Entity Resolution, Unsupervised Clustering

Received: 4 April 2010; Revised 19 May 2010; Accepted 3 July 2010

1. Introduction

According to recent U.S. Census Bureau reports, about 30% queries include person names. However, considering 100 million persons share only about 90,000 person names, a search result is a mixture of web pages of different people with the same name spellings. In general, this problem is known as *Mixed Entity Resolution Problem* for named entity search tasks on the Web (D. Lee, 2005). To demonstrate the need for a solution to mixed entities, let us present a real case drawn from Google, shown in Figure 1. In the search result, there exist a mixture of web pages of a professor at CMU, an actor, a hockey player, a historian, a Jazz guitarist, etc. who have the same name spellings of *Tom Mitchell*. There are 37 different *Tom Mitchell*s among top 100 ranked web pages as illustrated in Table 1. Furthermore, mixed entities commonly occur on the Web when we are searching information about a product by name. For instance, if a user searches for a product name such as *Oracle*, user also finds different web pages of Oracle Database, Oracle Audio, Oracle Academy, and so forth.

In this case, unlike traditional search engines, we focus on developing an effective system that identifies mixed entities such as person or product names as a query on the Web, and then displays its query result containing ranked groups, each of which contains URL links and corresponds to each different entity with the same description.

However, it is non-trivial to resolve mixed entities due to the following four challenges. First, since the number of clusters within top-*k* ranked web pages is not given a priori, we cannot take advantage of supervised clustering schemes such as *K*-means and *K*-spectral clustering. Second, skewed cluster sizes make it difficult to group web pages correctly. Next, the running time of clustering web pages should be instantaneous that users do not feel bored waiting search results for a long time. Finally, a set of clusters is required to be re-ranked. For instance, take a look at the name data set of *Tom Mitchell* in Table 1, where we observed that 92 top ranked web pages are grouped to 37 clusters of CMU professor, hockey player, historian, and so on. In the next step, the 37 clusters should be ranked in a certain order. In other words, the CMU professor cluster should be first ranked, and then the historian cluster is ranked, and so on. To cope with these challenges, we develop an effective framework for resolving mixed entities on the Web. In this paper, we propose a web service based interface, an unsupervised clustering, and several cluster ranking schemes. The system outline is shown in Figure 2. In particular, we devise an unsupervised clustering technique using *similarity propagation*.

The rest of this paper is organized as follows. We formally define mixed entity resolution problem. Then, we introduce an overview of our framework followed by discussion of our main ideas. Preliminary experimental results with name data sets are described next. Finally, some discussion and conclusion follow at the end.

Figure 1. Google search results showing that the ranked links of web pages associated with the search term of "Tom Mitchell"

2. Mixed Entity Resolution

As proof-of-concept, let us show two common, possible motivated examples as follows:

Example 1. An applicant *A* is applying for a job opening in a company. The search committee members in the company would like to understand *A* more than his resume. Thus they search for him in Google. When they query *A*'s name spellings in Google, it retrieves the web pages related to *A*. Unfortunately, there exist a number of web pages of different *A*s with the same name spellings. In addition, since *A* is not celebrity, his web pages are located far away from top ranked web pages in the search results. If we suppose that each result page of Google contains 10 links of web pages and the links related to *A*'s actual web pages are ranked between 90th and 100th positions, the committee members need to visit ten result pages to find *A*'s actual web pages.

Example 2. People often miss their class mates in high schools after their graduation. Indeed, to locate their friends, they attempt to search in Google using a query of their friends' name spellings. For instance, suppose that a friend name is "John Smith"' which is a common name in the country. Google would show us a mixture of web pages related to different "John Smith"s. Let us assume there are two *John Smith*s, where one is a carpenter and the other is a high school teacher. The number of web pages related to the school teacher is 98 and the web pages are located from the second position to the 99th position in 100 top ranked web pages retrieved from Google. On the other hand, the carpenter has 2 web pages and the web pages are ranked in the first position and the 100th position.

Formally, mixed entity resolution problem is defined as follows[1]:

> Given a set of mixed entities $E=\{e_1,...,e_p,...,e_q,...,e_N\}$ with the same name description d, group E into K disjoint clusters $C=\{c_1,...,c_K\}$ such that entities $\{e^i_p,...,e^i_q\}$ within each cluster c_i belongs to the same group.

Clustering is the key part for resolving mixed entities. Consequently, we view the problem as an unsupervised clustering problem. We conjectured that the majority of input pages map to a single individual, although there are a few cases that are assigned to multiple individuals sharing the same name. Hence, we view the problem as a *hard* clustering which assigning input pages to exactly one individual cluster so that the produced clusters are not overlapped.

Hard clustering algorithms can be classified as either a partitioning or a hierarchical clustering. Hierarchical agglomerative clustering approach generates a series of nested clusters by merging simple clusters into larger ones, while partitive methods try to find a pre-specified number of clusters that best capture the data. For instance, a prior knowledge of the probable number of clusters must be required in *K*-means and *K*-way spectral clustering algorithms.

Since the correct number of clusters is not known *a priori* in our problem, Hierarchical Agglomerative Clustering (HAC) algorithms, rather than partitive clustering methods, are employed as a solution to our problem. However, hierarchical clustering methods are not able to reallocate entities which are plausible to be poorly classified in the early stages of text analysis. In this paper, we study an unsupervised clustering method that is more accurate than HAC algorithms.

[1]In our work, E is a collection of web pages and e_p stands for p-th web page and d denotes person or product name.

3. Main Proposal

Figure 2 illustrates the overall architecture of our system. To search for a product name (e.g., Oracle), we use Google web service framework in which Google web service server provides us with a list of top-*k* ranked web pages which are associated with the product name. Then, we tokenize top-*k* ranked web pages and each web page is represented as a vector using TF/IDF weighting scheme and the similarity of each pair of vectors are computed by TF/IDF cosine similarity measure. Subsequently, the vectors are grouped into a set of clusters in terms of our *similarity propagation* based unsupervised clustering method. Finally, *K* clusters are ranked by one of our cluster ranking algorithms followed by the search result.

3.1 Web Service Based Interface

As shown in Figure 2, our web service client to Google (Google Web APIs) uses a keyword search in the Google search engine. Since the Google web service supports Simple Object Access Protocol (SOAP), which is a technology to allow for Remote Procedure Call (RPC) over the web, the client program creates a SOAP request message that contains a person name entered by a user, and then sends it to the Google's web service server. After the client receives a SOAP response message from the server, it parses the SOAP response, and then extracts the top-*k* links. According to the recent study (B. Jansen, 2003), the majority of people only tend to look into the first returned page (i.e., 10 links) of Google. Thus, we focus on at most *k=100* links which would be the URLs of web pages that contain the keyword.

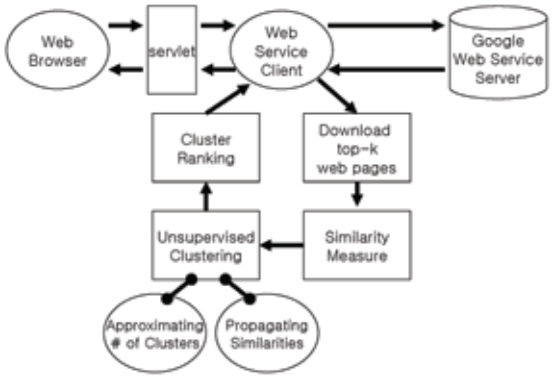

Figure 2. System Architecture

3.2 Similarity Measure

Web pages corresponding to top-*k* links are downloaded in our system. Let e_p, t_i, and T be p-th web page, the i-th term in e_p, and the total number of distinct terms in the top-*k* web pages, respectively. w_{t_i} is the weight value which is associated with the pair (t_i,e_p), and further document vector $v_{e_p} = (w_{t_1}, w_{t_2}, ...,w_{t_T})$. First, to weight each term t_i in e_p, $w_{t_i} = \frac{F(t_i)}{\max_{t_l} F(t_l)} \times \log\left(\frac{N}{N_{t_i}}\right)$, where $F(t_i)$ is the raw frequency of term t_i in e_p; N is the total number of documents in the system; and N_{t_i} is the number of documents in which term t_i appears. Then, the similarity between v_{e_p} and v_{e_q} is estimated as $sim(v_{e_p}, v_{e_q}) = \frac{v_{e_p} \cdot v_{e_q}}{|v_{e_p}| \times |v_{e_q}|} = \frac{\sum_{t=1}^{T} w_{t_i,e_p} \times w_{t_i,e_q}}{\sqrt{\sum_{i=1}^{T} w^2_{t_i,e_p} \times \sum_{i=1}^{T} w^2_{t_i,e_q}}}$.

3.3 Approximating Number of Clusters

Based on our similarity scheme, top-k web pages can be represented as a graph G, where p-th web page denotes node e_p, and the edge weight between nodes e_p and e_q is the TF/IDF cosine similarity between e_p and e_q. In next step, we need an unsupervised clustering method to gather similar entities together. Since the unsupervised clustering generally shows poor performance, we devised an algorithm estimating the number of clusters shown in Figure 3. The main idea for this algorithm is to gradually disconnect several edges based on the connectivity between nodes and analyze the patterns of segmented subgraph sequences. Assume N_{μ_i} and $N_{\mu_{i+1}}$ are number of subgraphs with μ_i and μ_{i+1}, respectively. Then, we assumed that the number of clusters are assumed to be near when the difference between N_{μ_i} and $N_{\mu_{i+1}}$ is drastically changed. We observed three different types of graph segment sequences with an incremental threshold value μ such as gradually increasing, gradually decreasing, and staying approximately as a constant.

In the gradually increasing sequence (convex form), the sequence reaches the maximum difference when the difference starts to decrease. For the gradually decreasing sequence (concave form), the sequence reaches the minimum difference when the difference starts to increase. Based on this, 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 form, the minimum difference when it shows convex form, and the average of these two when it stays approximately as a constant (linear form).

Input: Graph G
Output: # of clusters
  For {$\mu = 0.01;\mu \le 1$: $\mu = \mu+0.01$}
    Generate a subgraph G_i by removing
    links s.t. their link weights $\le \mu$
    $numClus_i$ = # of disconnected graph segments G_i
    E=(m, $numClus_i$)
  End For
  Based on the E sequences, classifies as
  Type I (concave), Type II (convex), Type III (linear)
  Find the maximum and minimum difference between
    $numClus_i$ and $numClus_{i+1}$ in E
  If graph segments {G_i} are in Type I
    Return maximum difference $numClus_i$ as cluster #
  Elself graph segments {G_i} are in Type II
    Return minimum difference $nimClus_{i+1}$ as cluster #
  Else graph segments {G_i} are in Type III
    Return average of maximum and minimum difference

Figure 3. Approximation of Cluster Numbers

3.4 Similarity Propagation

We approximate the number of clusters (K) of the given input data. Then, we are able to apply K-centers clustering method. However, the disadvantage of the K-centers clustering method is that the clustering results is affected by the initial selection of centroids at random. In case of incorrect selection of K centroids, it would have poor clustering results. To improve the weak point of the K-centers method, we consider all nodes as potential centroids. Furthermore, we propagate pair-wise similarities along links of a graph G until K good centroids emerge.

The similarity sim (v_{e_p} ,v_{e_q}) stands for how well v_{e_p} is suited to be the centroid vector for v_{e_p} , defined as sim (v_{e_p} , v_{e_q}) = -| v_{e_p}-v_{e_q} |. To determine which vectors are centroids and which centroid each vector belongs to, two kinds of messages are exchanged between vectors. One is **Responsibility** R (v_{e_p} , v_{e_q}) and the other one is **Availability** A (v_{e_p} , v_{e_q}). More specifically, R (v_{e_p} , v_{e_q}) indicates how well suited v_{e_q} is to work as centroid for v_{e_p} considering other centroids for v_{e_p}. R (v_{e_p} , v_{e_q}) is sent from v_{e_p} to v_{e_q}. In contrast, A (v_{e_p}, v_{e_q}) reflects how appropriate it would be for v_{e_p} to choose v_{e_q} as its centroid vector, sending from centroid v_{e_q} to v_{e_p}. Initially, A (v_{e_p}, v_{e_q}) = 0. The responsibilities are computed by $R(v_{e_p}, v_{e_q}) = sim(v_{e_p}, v_{e_q}) - \max_{v'_{e_q} \neq v_{e_q}} \{A(v_{e_p}, v'_{e_q}) + sim(v_{e_p}, v'_{e_q})\}$

The availability updates to see if each candidate centroid vector would make a good centroid defined as $A(v_{e_p}, v_{e_q}) = \min\left\{0, R(v_{e_q}, v_{e_q}) + \sum_{v'_{e_p} \in \{v_{e_p}, v_{e_q}\}} \max\{0, R(v'_{e_p}, v_{e_q})\}\right\}$ The self-availability A (v_{e_p} ,v_{e_q}) is updated differently in terms of $A(v_{e_q}, v_{e_q}) = \sum_{v'_{e_p} \neq v_{e_q}} \max\{0, R(v'_{e_p}, v_{e_q})\}$. This self- availability indicates that v_{e_q} is a centroid, based on the positive responsibilities sent to candidate centroid v_{e_q} from other vectors. Using the above functions, messages are exchanged between vectors until a high quality set of centroids and corresponding clusters gradually converges. In other words, for vector v_{e_p}, the value of v_{e_q} maximizing A (v_{e_p}, v_{e_q}) + R (v_{e_p}, v_{e_q}) identifies the vector that is the centroid for or the vector itself as the centroid. The time complexity of the similarity propagation is O($i \times K \times N$), where i, K, and N are the number of iterations, centroid vectors, and entity vectors, respectively. Details of the method are described in (Frey, 2007).

3.5 Cluster Ranking Algorithms

As the result of our clustering method using similarity propagation, relevant web pages are clustered to the same group. For instance, suppose there are four different individual persons with the same name spellings, shown in Table 1. In the example, the total number of web pages related to "Tom Mitchell" is 13, where four web pages (i.e., e_1, e_2, e_3, and e_4), three ones (i.e., e_5, e_6, and e_7), five ones (i.e., e_8, e_9, e_{10}, e_{11}, and e_{12}), and one web page (e_{13}) are associated with professor at CMU, reporter at CNN, musician at Shady record company, and minister at Kansas city, respectively. Let us assume that web pages are ranked by PageRank scores like the third column in Table 1. The ranking order of web pages in terms of Google PageRank is e_1, e_5, e_{10}, e_6, e_7, e_9, e_8, e_4, e_{12}, e_2, e_{11}, e_3, and e_{13}.

Cluster Label	Web Page ID	Google PageRank
Tom Mitchell Professor CMU	e_1 e_2 e_3 e_4	1 10 12 8
Tom Mitchell Reporter CNN	e_5 e_6 e_7	2 4 5
Tom Mitchell Musician Shady Records	e_8 e_9 e_{10} e_{11} e_{12}	7 6 3 11 9
Tom Mitchell Minister Kansas City	e_{13}	13

Table 1. An example of ranking clusters

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 *ranking* clusters.

This is a considerably challenging issue in this paper. For instance, note the four web pages of "Tom Mitchell" at CMU. e_1 is firstly ranked by Google PageRank algorithm, while the rest pages are mostly located in bottom -- e_2 (10-th), e_3 (12-th), and e_4 (8-th). In this case, it is hard to determine which position the cluster (labeled as CMU/Prof) should be ranked in 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 when we attempt to rank a set of clusters*. For this, we consider three different methods as follows:

- Consider the *relative ranking positions* of web pages per cluster which is defined as $ClusterRank_j = \frac{\sum_{i \in cluster_j} PageRank_i / N}{N_{cluster_j}}$ where $Cluster_j$ is the number of pages in cluster j and N 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 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 $ClusterRank_j = \min_{i \in Cluster_j} PageRank_i$. For instance, the cluster of "CMU/Prof" is ranked in the first position due to the ranking position of e_1. In other words, ClusterRank(CMU/Prof) = 1. 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 $ClusterRank_j = median\{Cluster_j\}$ As an example, ClusterRank(Shaby/Mus) = 7 due to PageRank(e_{10}) = 3, PageRank(e_9) = 6, **PageRank**(e_8) = **7**}, PageRank(e_{12}) = 9, and PageRank(e_{11}) = 10. Finally, the clusters are re-ordered by the *ClusterRank* scores in the ascending order.

Considering that our ranking algorithms compute the ranks of clusters based on document ranks by Google PageRank, the result lists of our ranking algorithms would be analogous to Google's standard output. However, we expect that the result lists of our ranking algorithms would provide users with better presentation. For instance, please see Tom Mitchell at CMU in Table 1, in which e_1, e_2, e_3, and e_4 pertaining to the CMU professor are ranked in 1-st, 10-th, 12th, and 8-th. In general, Google standard output is a set of pages containing top-*10* documents. Thus users are able to see three documents in the first result page, and the other page in the second result page. On the other hand, in terms of our relative ranking algorithm, e_1, e_2, e_3, and e_4 are ranked in 5-th, 7-th, 8th, and 6-th, and all the documents appear in the first result page containing top-*10* documents. In the end, users are able to search for all documents (related to Tom Mitchell at CMU) once.

4. Experimental Results

We used Google APIs (Google Web APIs) to implement Google web service client and server programs. We also measure similarities of documents using TF/IDF cosine similarity of SecondString (SecondString: Open source Java-based package of Approximate String-Matching). For the similarity propagation based clustering method, we used the C-programming code of (Frey, 2007) in public. All experimentations were done on 4 × 2.6 Ghz Opteron processors with 32GB of RAM.

4.1 Testing Environment

Data sets. For validation, we have used two data sets from real examples on the Web. The *person name* is a test case using the 1,085 web pages that (Bekkerman, 2004) used. In 2004, Bekkerman extracted 12 personal names from Melinda Gervasio's email directory. Then, 100 top-ranked web pages of each name were retrieved from Google, and cleaned and manually labeled by authors. The resulting data set consists of 1,085 web pages, 187 different persons, and 420 relevant pages. Table 2 shows the statistics of the data set. For instance, when "Tom Mitchell" is issued as a query to Google, 92 web pages are retrieved. Among these 92, there are 37 namesakes to "Tom Mitchell." For example, among 92 web pages, "Tom Mitchell" appears as musicians, executive managers, astrologist, hacker, and rabbi -- 32 different kinds. That is, a set of 32 individual persons are mixed since they all have the same name description of "Tom Mitchell". Similarly, the *product name* is the other test case as illustrated in Table 2.

Keyword Type	Name	N	K
Person	Adam Cheyer	97	2
	William Cohen	88	10
	Steve Hardt	81	6
	David Israel	92	19
	Leslie Pack Kaelbling	89	2
	Bill Mark	94	8
	Andrew McCallum	94	16
	Tom Mitchell	92	37
	David Mulford	94	13
	Andrew Ng	87	29
	Fernando Pereira	88	19
	Lynn Voss	89	26
Product	Oracle	25	8
	Sun	25	14
	Trojan	25	9

Table 2. Characteristics of name data set (N: # of top-k pages and K: # of clusters as true solution)

Generate Similarity Graph. To create a terminology document matrix A, we used TMG (D. Zeimpekis, 2006) software package with spamming, and dropped common words using dictionary, and applied normalization. Each A_{ij} element indicates the term frequency (TF) multiplied by inverse document frequency (IDF) of terminology t_i in the document d_j as in $A_{ij} = TF_{ij} \times IDF_{ij}$. Then, we generate document-document matrix G by multiplying document term matrix, A^T, with term document matrix, A, as in $G = A^T \times A$. The $G(i,j)$ element in the matrix indicates the similarity value of two documents d_i and d_j. G_{ij} element holds sum of multiplications of d_{ik} and d_{jk} values for each terminology t_k as in $G = A^T \times A(i,j) = \sum d_k \times d_k$ as described in (M. Berry M. B., 2005).

Evaluation Metrics. To evaluate competitive clustering methods, we use *rand index* R_1 (Rand, 1971). Given a set of N entities and two clusters X and Y, $R_1 = (a+b)/(a+b+c+d)$, where a : number of pairs of elements that are in the same set in X and Y; b : number of pairs of elements that are in different sets in X and Y; c : number of pairs of elements that are in the same set in X and in different sets in Y; and d : number of pairs of elements that are in different sets in X and in the same set in Y. If R_1 is closed to 1, the two partitions X and Y are the same. For instance, suppose that X is the predefined cluster sets and Y is the cluster sets reported by either HAC algorithms or our similarity propagation based clustering technique. Let $X = \{(1,2,3),(4,5,6)\}$ and $Y = \{(1,2),(3,4,5,6)\}$. Then, $a = \{(1,2),$

Name	Solution Cluster	Similarity Propagation			Hierarchical Clustering		
		Cluster #	Relative Error	Rand Index	Cluster #	Relative Error	Rand Index
Adam Cheyer	2	9	3.50	0.17	39	18.5	0.11
William Cohen	10	6	0.40	0.46	59	4.90	0.42
Steve Hardt	6	9	0.50	0.43	45	6.50	0.41
David Israel	19	12	0.37	0.70	50	1.63	0.76
Leslie Kaebling	2	2	0.00	0.50	59	28.5	0.03
Bill Mark	8	7	0.13	0.64	58	6.25	0.60
Andrew McCallum	16	12	0.25	0.67	67	3.19	0.65
Tom Mitchell	37	27	0.27	0.88	66	0.78	0.94
David Mulford	13	31	1.38	0.65	52	3.00	0.65
Andrew Ng	31	37	0.19	0.82	24	0.23	0.67
Fernando Pereira	19	15	0.21	0.74	33	0.74	-.67
Lynn Voss	26	26	0.00	0.82	40	0.54	0.86
Sun	14	9	0.36	0.90	12	0.14	0.92
Oracle	8	5	0.38	0.61	8	0.00	0.66
Trojan	9	4	0.56	0.80	6	0.33	0.76

Table 3. Experimental results. Relative error shows huge difference between two algorithms

(4,5), (4,6), (5,6)} = 4. b = {(1,4), (1,5), (1,6), (2,4), (2,5), (2,6)} = 6. c = {(1,3), (2,3)} = 2. d = {(3,4),(3,5),(3,6)}=3. Thus, $R = (4+6) / (4+6+2+3) = 0.67$. Another metric we are using to measure the performance is relative error which is defined as the difference between the actual and predicted number of clusters divided by the actual number of clusters. For example, if the algorithm generates 4 clusters but the true solution is 5, then the relative error is 0.2. The relative error shows how close the clustering results to the true solutions.

5. Results

Correctness. Table 3 shows the experimental results of two algorithms on our name data sets. For the hierarchical clustering algorithm, we manually try several different cut off values and choose the best results. However, hierarchical clustering shows wide margin of errors in predicting the number of clusters with our name data set. Similarity propagation with our cluster predicting heuristic described in the previous section shows almost ten times better result in terms of relative error. In addition, hierarchical clustering requires pairwise distances to build up the linkage tree which requires $O(n^3)$ number of computations and also needs $O(n^2/2)$ amount of memory space to store the linkage data where n is the number of documents. On the other hands, similarity propagation based on sparse matrix requires only $O(nnz^2)$ computations and $O(nnz)$ memory space where nnz is number of related document pairs.[2]

As a conventional method to measure the quality of clustering results, we used rand index as described in the previous section. The experimental results show that our similarity propagation clustering shows 4% better performance than hierarchical clustering algorithm. Considering the rand index becomes much smaller if the number of clusters is increasing, the 4% performance difference between two algorithms is significant. These two algorithms show the similar difference in other methods such as f-measure and jaccard index.

Cluster Ranking. Figure 4 illustrates the screen-shot of our three ranking algorithms. To search for *Trojan* in our system, we choose top-25 ranked web pages from Google. According to our manual inspection, there exist nine clusters. In other words, there is a mixture of nine distinct Trojan web pages in Google -- *Trojancondoms.com*, *Los Angeles Times*, *Trojan horse*, *Daily Trojan*, *University of Southern California Trojans Official Athletic Site*, *Trojan Minor Planets* (http://www.harvard.edu), *Trojan Battery Company*, *Troy University Trojans Athletics Official Site* (TroyTrojans.com), and *Trojan Women by Euripides* (http://classics.mit.edu). Furthermore, we can summarize the web pages with regard to Trojan which are condoms, university athletic sites, computer virus protecting software, ancient stories related to Troy Trojan, blogs, and so on.

According to Google PageRank, *Trojancondoms.com* and *Los Angeles Times* are top-3 web pages. As shown in Figure 4, such top-3 web pages are located in the same ranking positions as Google PageRank in both median and relative ranking algorithms. In particular, median ranking algorithm is considerably similar to relative ranking algorithm, while both are slightly different from highest ranking algorithm. Since highest ranking algorithm ranks a cluster c_i in terms of only one web page with the *highest ranking position* among the other web pages within ci, the ranking positions of clusters are more variants than the other ranking algorithms. As shown in Figure 4, *Trojancondoms.com* is one cluster which comprises web pages associated with condoms, and *Trojan Battery Company* is another cluster which includes a set of web pages regarding computer virus protecting software. By and large, 25 web pages are correctly grouped to a set of clusters.

6. Related Works

Bekkerman et. al. proposed two algorithms to disambiguate web appearances of people in a social network in their paper (R. Bekkerman, 2005). One is based on link structure of web pages and another is using multi-way distributional clustering method. Their algorithms show improvement in the aspect of accuracy. Minkov et. al. used lazy graph walk algorithm to disambiguate names in email documents in their paper (E. Minkov, 2006). The authors 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 (A. Banerjee, 2007). Different types of entities are simultaneously clustered based not only on their intrinsic attribute values but also on multiple

[2]Generally, the number of nonzero elements (nnz) in sparse matrix is way smaller than n^2.

(a) Highest Ranking (b) Median Ranking (c) Relative Ranking

Figure 4. Results of three ranking algorithms

relations between entities. On et al. introduced multi-level graph partitioning scheme to address the scalable issue of name disambiguation problem on both bibliographic and information retrieval domains (B. On, 2007). Lee. et. al. showed algebraic approach to solve name disambiguation problem using SVD and NMF in (I. Lee, 2009).

On the other hand, to estimate the number of clusters, a number of approaches have been proposed. In particular, recent leading studies are Gap Statistic (R. Tibshirani, 2001) and Clest (S. Dudoit, 2002). However, such methods have been developed to address the problem in which the cluster sizes are well distributed. In addition, they focus on estimating a small number of clusters (e.g., at most 2 or 3 clusters). In author awareness, this is the first paper to provide a systematic approach in solving a mixed entity resolution problem. In addition, we analyzed the name disambiguation problem characteristics, proposed an unsupervised clustering algorithm to solve extremely skewed clustering problem, and proposed a prototypical system. Our experiment results show some promising in choosing a proper algorithm based on the problem and computational environments.

7. Conclusion and Future Work

In a nutshell, we formalized the mixed entity problem which commonly appears on the Web. Then, we developed a practical system for resolving mixed entities such as person or product names for name search tasks. For development of such a system, we introduced web service based interface. In addition, since a prior knowledge of the probable number of clusters is unknown, we presented an unsupervised clustering schemes based on similarity propagation that outperforms the existing well-known Hierarchical Agglomerative Clustering algorithms. Finally, we proposed three ranking algorithms for arranging the resulted clusters in an appropriate order. In practice, our proposal can be used as *name search* in Google[3]

For our future direction, the scalability of the algorithm is an ongoing problem. Note that we focus on top-*k* web pages retrieved from Google. In this paper, our system correctly group only top-*k* web pages to a set of clusters. As the *k* value is increased, our clustering schemes suffer from scalable problem. To address this challenging problem, we are working on unsupervised clustering methods based on multi-level graph partitioning approach. In addition, it is infeasible for every clustering method to correctly (perfectly) cluster web pages at all. To cope with this practical issue, we will apply the concept of feedback and investigate semi-clustering problem (induced by users' feedback) in our future work.

[3]Currently Google provides us with a variety of keyword searching services -- normal search, advanced search, image search, and so on

References

[1] Banerjee, S. B. (2007). Multi-way Clustering on Relation Graphs. *SIAM Data Mining.*

[2] Al-Sultan, K. (1996). Computational experience on four algorithms for the hard clustering problem, *Pattern Recognition Letter*, 17 (3) 295-308.

[3] Jansen, A. S. (2003). An Analysis of Web Documents Retrieved and Viewed, *In: Int'l Conf. on Internet Computing.* Las Vegas, USA.

[4] On, D. L. (2007). Scalable Name Disambiguation using Multi-level Graph Partition. *SIAM Data Mining.*

[5] Bekkerman, R. (n.d.). *Name Data Set*. Retrieved from http://www.cs.umass.edu/ronb

[6] Chua, F. (2009). Dimensionality Reduction and Clustering of Text Documents. Singapore Management University.

[7] Lee, B. O. (2005). Effective and Scalable Solutions for Mixed and Split Citation Problems in Digital Libraries, *In:* ACM SIGMOD Workshop on Information Quality in Information Systems (IQIS). Baltimore, USA.

[8] Zeimpekis, E. G. (2006). TMG: A MATLAB toolbox for generating term document matrices from text collections.

[9] Elmacioglu, Y. T. (2007). PSNUS: Web People Name Disambiguation by Simple Clustering with Rich Features, *In*: Int'l Workshop on Semantic Evaluation (SemEval), (p. 268-271). Prague, Czech Republic.

[10] Minkov, W. C. (2006). Contextual Search and Name Disambiguation in Email using Graphs. SIGIR.

[11] Frey, B. D. (2007). Clustering by Passing Messages Between Data Points, *Science*. 315.

[12] Google Web APIs. (n.d.). Retrieved from http://www.google.com/apis

[13] Heath, M. (2002). Scientific Computing: An Introductory. New York: McGraw Hill.

[14] Lee, B. O. (2009). Algebraic Algorithms to Solve Name Disambiguation Problem. International Conference on Data Mining. Las Vegas, USA.

[15] Han, M. K. (2001). Spatial clustering methods in data mining: A survey. Taylor and Francis.

[16] Jain, M. F. (1999). Data clustering: A review, *ACM Computing Surveys*, 31.

[17] Jaleel, A. M. (2006). Last-level cache (llc) performance of data-mining workloads on a cmp{a case study of parallel bioinformatics workloads. 12th HPCA.

[18] Yeung, W. R. (2001). Principal Component Analysis for Clustering Gene Expression Data. Bioinformatics, 17(9) 763-774.

[19] Keyes, D. (2004). Retrieved from Terascale Optimal PDE Simulations (TOPS) Center: http://tops-scidac.org

[20] Kaufman, P. R. (1990). Finding Groups in Data: An Introduction to Cluster Analysis. Wiley.

[21] Berry, M. B. (2005). Understanding Search Engines. SIAM Press.

[22] Berry, S. D. (1995). Using linear algebra for intelligent information retrieval, *SIAM Review*, 37. 573-595.

[23] MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations, *In*: Fifth Berkeley Symposium on Mathematical Statistics and Probability.

[24] Slonim, N. F. (2002). Unsupervised Document Classification using Sequential Information Maximization. SIGIR.

[25] Baeza-Yates, B. R.-N. (1999). Modern Information Retrieval. Addison Weseley.

[26] Bekkerman, A. M. (2005). Disambiguating Web Appearances of People in a Social Network. Int'l Conf. on World Wide Web (WWW).

[27] Duda, P. H. (1973). Pattern Classification and Scene Analysis. Wiley and Sons.

[28] Tibshirani, G. W. (2001). Estimating the Number of Clusters in a Dataset via the Gap Statistic. J. R. Statist. Soc. B, 411-423.

[29] Rand, W. (1971). Objective Criteria for the Evaluation of Clustering Methods, *Journal of American Statistical Association,* 66. 846-850.

[30] Dudoit, J. F. (2002). A Prediction-based Resampling Method for Estimating the Number of Clusters in a Dataset. Genome Biology, 3(7).

[31] SecondString: Open source Java-based package of Approximate String-Matching. (n.d.). Retrieved from http://secondstring.sourceforge.net/

[32] WEPS: Searching information about entities in the web. (n.d.). Retrieved from http://nlp.uned.es/weps/