Social Network Analysis on Name Disambiguation and More

Byung-Won On
Department of Computer Science
The University of British Columbia
bwon@cs.ubc.ca

Abstract

Name variants are ubiquitous in real world due to typographical errors (e.g., "Forschungszentrum JUlich" vs. "Forschungszentrum Julich"), abbreviated, imcomplete, or missing information (e.g., "R. E. Ellis" vs. "Randy E. Ellis"), lack of standard name formatting convention (e.g., "Spike Jonze" vs. "Jones, Spike"), and their combinations. In this paper, we project this name disambiguation problem to graph representation, and then analyze graphs using social network analysis. In particular, we used real duplicate name entities that we manually verified from ACM digital library. Then, using various string similarity metrics and additional information (i.e., co-author names, titles, and veneus), we analyze the effectiveness of string similarity metrics and additional information based on social network analysis. Through our experimental validation, name disambiguation problem can be analyzed in graphical, visual manner.

1 Introduction

Since a person name can exhibit multiple variants in real world, searching for a person name is considerably challenging in bibliographic digital libraries and yellowpages. The name search problem is a specialized problem of Entity Resolution (ER) in database and data mining communities. If users, searching for a name, enter a variant not matched representative names indexed in the system, they may obtain *incorrect* search results. To avoid this problem, systems, as a pre-processing step, regularly peform *name disambiguation process* from the crawled data. In order to demonstrate the need for a solution to the name disambiguation task, let us present the following real case drawn from the ACM Portal.

Example 1. Figure 1 is a screen shot of the ACM Portal, which contains the list of author names who have ever published an article in ACM-affiliated conferences or journals. In particular, Figure 1 shows author names whose last names are either "Ullman" or "Ullmann". Note that the name of the renowned computer scientist, "Jeffrey D. Ullman" at Stanford University, appears as ten variants, *incorrectly*. For instance, "J. D. Ullman" in the list is in fact the same person as "Jeffrey D. Ullman", however they are treated as different scholars. In such a case, users searching for all the articles of "Jeffrey D. Ullman" will get only citations of "Jeffrey D. Ullman". Any bibliometrical study would underestimate the impact of the author "Jeffrey D. Ullman", splitting his fare share into ten variants incorrectly.

As the solution, to identify variant names, we treat additional information (i.e., co-author names, titles, and venues) related to an author name as a "string", and measure all pairwise string similarities using a certain similarity function. If $sim(x, y) \geq \theta$, x and y are name variants. Then, we generate a duplicate entity graph with variants identified by our approach. Note that such a duplicate entity graph is a semantic graph. That is, the graph is not formed *syntatically* by name spellings but rather *semantically* by the similarities of pair-wise nodes.

In this paper, we project name disambiguation problem to graph representation. Then, we strive to analyze the duplicate entity graph using social network analysis that provides effective methods for analyzing semantic graphs in order to figure out name disambiguation problem in graphical and visual flavor. For details, if two nodes are connected to each other in the duplicate entity graph, they are likely to be variant names. Observing the topological features (i.e., degree, distance, betweenness, and clustering coefficient distribution) of the graph, we can be aware of name disambiguation problem more precisely. That is, investigate if the graph is random, regular, scale-free, or small world network. Note that there exist on average 3 variant names in our data set that we manually collected from ACM digital library. Therefore, we have our own point of view that

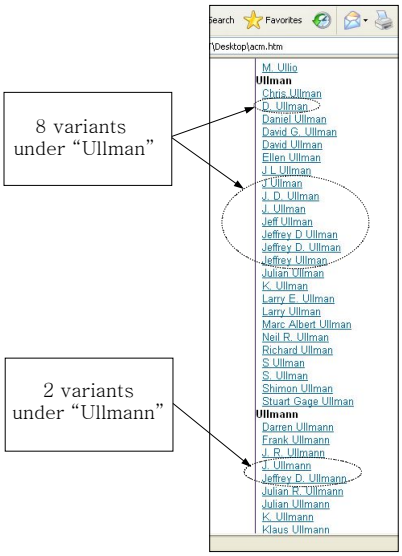


Figure 1. Screen-shot of author index for "Ull*" in the ACM Portal. Note that the citations of "Jeffrey D. Ullman" appear as eight variants under "Ullman" and two variants under "Ullmann."

if a string similarity method finds variant names well, the duplicate entity graph should show the topological features of random graphs. Otherwise, for instance, if it is a scale-free network, the used similarity metric and context information are not effective to identify redundant entities. In addition, we further examine that the duplicate entity graph is affected by different types of string similarity measures as well as by different attributes.

This paper is organized as follows. Section 2 briefly introduces the overview of problem definition, solution, and related work. Section 3 describes our name disambiguation analysis framework. Section 4 reports our experimental set-up and results. Finally, Section 5 concludes and discusses our future work.

2 Preliminaries

2.1 Name Matching Problem Definition

We formally define the *Name Matching* problem – Given two *long* lists of entities (i.e., author names), X and Y, for each entity x ($\in X$), find a set of entities, y_1, y_2, ..., y_n ($\in Y$) such that both x and y_i ($0 \leq i \leq n$) are variants of the same entity. The solution is to treat additional information (e.g., titles) of an author name spellings as a "string", and perform all pair-wise string similarities using a certain similarity function. If $sim(x, y) \geq \theta$, x and y are name variants.

2.2 Related Work

2.2.1 String Similarity Metrics

As a main solution to the name matching task, various string similarity metrics have been used to effectively identify duplicates refering to the same name entity in the context that string distance measures are most useful for matching problems with little prior knowledge or unstructured data [4]. To address the name matching task, various string similarity metrics have been proposed, and by and large, string similarity metrics are categorized into four classes as follows:

- **Character-based Similarity Metrics** relies on character comparison to handle typographical variations. Edit Distance [13, 12, 15, 19], Affine Gap Distance [23, 3, 19], Smith-Waterman Distance [22], and Jaro [10] are such similarity metrics. Given two strings s_1 and s_2, Edit Distance estimates the minimum number of edit operations to transform s_1 into s_2, where three types of edit operations are *character insertion*, *deletion*, and *substitution*.

- **Token-based Similarity Metrics.** As a main error type of strings, string tokens are often rearranged. For instance, "John Smith" versus "Smith, John". In such case, edit distance metrics are not the effective method to identify the duplicates. On the other hand, using multisets of words of two strings, token-based similarity metrics, such as Jaccard, Monge-Elkan [14], and n-grams with TF/IDF [7], work well for alternated string tokens.

- **Phonetic Similarity Metrics** are designed to address redundant cases that the representation of two strings (e.g., "Cajun" versus "Kageonne") are different while they are phonetically similar each other. First, Soundex [20, 21] retains the mapping information to assign character alphabets to phonetically similar groups of consonants. Moreover, the NYSIIS system and Oxford Name Compression Algorithm (ONCA) [6] take the position of vowels into account. As a better method than Soundex, Metaphone [17] and Double Metaphone [18] use 16 consonant sounds to cover many English and non-English words.

- **Numeric Similarity Metrics.** Rather than strings, this problem is to detect numbers, treated as strings, with similar values. Koudas et al. [11] consider the distribution and type of the numeric data. Agrawal et al. [1] extend concept of cosine similarity for numeric data.

Discussion. Both phonetic and numeric similarity approaches are not related to our name matching problem.

Thus, we do not take such similarity measures into account in this paper.

Furthermore, on the task of matching name entities, no string similarity metric provides concrete accuracy in a variety of data. For instance, Edit distance method works effectively for capturing spelling errors or insertions and deletions of short words. For instance, as for two author names "R. Agrawal"and "R. Agrrawal", the edit distance $= 1$. On the other hand, edit distance metrics do not work well when the compared names include variations of words order or insertions and deletions of long words. For example, "R. Agrawal" versus "Agrawal, Rakesh". Similarly, token-based similarity metrics have the strong point in comparing strings that contain insertions and deletions of common words or variations of word order. However, such methods can mismatch duplicates with spelling errors, compared to edit distance functions [7]. Thus, we use effective methods in both character and token-based similarity metrics.

3 Our Name Disambiguation Analysis Framework

First, we apply a string similarity measure to our input data, and then we will obtain redundant name entities. Next, we represent the redundant data as a duplicate entity graph. As a final, comparatively examine theoretical descriptions of the relationships in the duplicate entity graph.

3.1 Graph Formation

Given a list of name entities as an input, to obtain duplicate entities, apply a string similarity measure, such as Jaccard or Cosine similarity, to all pairs of name entities to see if each pair of name entities is identicial. For instance, given two author names "Jeff Ullman" and "Peter Ullman", $sim_{Jaccard}$(context("Jeff Ullman"), context("Peter Ullman")) $= 0.33$. Since the $sim_{Jaccard}$ value $< \theta (= 0.6)$, where θ is the pre-determined threshold value. We consider that the two names are different. Thus, the edge between "Jeff Ullman" and "Peter Ullman" is not created in the duplicate entity graph.

3.2 String Similarity Methods

In this scheme, the similarity between two author names are measured by the "similarity" between their context information. In our framework, *context information* is citations. In particular, each citation consists of three sets of attributes: (1) co-authors (2) titles (3) venues. That is, to measure the similarity between "John Smith" and "Smith, J.", instead of computing the sim("John Smith", "Smith, J."), we compute the sim(co-authors("John Smith"), co-authors("Smith, J.")).

Name	Description
x, y	co-author names
T_x	all tokens of the co-author x
C_x	all characters of x
$CC_{x,y}$	all characters in x common with y
$X_{x,y}$	# of transpositions of char. in x relative to y

Table 1. Terms in a set of co-authors.

3.2.1 Jaccard

Among many possible token-based similarity measures, we used Jaccard similarity that were reported to give a good performance for the general name matching problem in [4]. We briefly describe the metric below. Using the terms of Table 1, the metric can be defined as follows:

- $\mathbf{Jaccard}(\mathbf{x}, \mathbf{y}) = \frac{|T_x \bigcap T_y|}{|T_x \bigcup T_y|}$;

3.2.2 Edit distance

Given two names n_1 and n_2, edit distance is the cost of best sequence of edit operations that convert n_1 to n_2. Typical edit operations are character insertion, deletion, and substitution, and each operation must be assigned a cost. For our experiment, we make use of Levenshtein distance [5].

3.2.3 Cosine Similarity

In this approach, instead of using string similarity, we use vector similarities to measure the similarity of the context information. We model a set of attribute (e.g., co-author names) as vectors in the vector space, each dimension of which corresponds to a unique author name appearing in the citations in the block. For example, suppose we have a block that has three groups of citations, one for "D. Lee", another for "D. W. Lee" and the other for "D. Y. Lee". In the first group, suppose we have five papers, each coauthored by "D. Lee" with one or more of "J. Kang", "P. Mitra", and "T. Han". Further suppose among the five papers "D. Lee" coauthored three times with "J. Kang", four times with "P. Mitra", and once with "T. Han". Similarly, suppose we have 10 papers in the second group, in which "D. W. Lee" coauthored five times with "J. Kang", seven time with "P. Mitra", and three times with "W. Chu". In the last group, supposed we have seven papers in which "D. Y. Lee" coauthor red three times with "P. Chopra", five times with "J. Xu", and once with "J. Kang".

The total number of unique coauthor names in the block is 6 except the three "D. Lee" name variants. In order to determine if the three name variants are indeed referring to the same person, we model the the coauthor information for each variant as a vector and compute the distances between the vectors. The resulting vectors have 12 dimensions, each

of which corresponds to one of 6 unique coauthor names appearing in the block. For example, for the first group of citations for "D. Lee", we have a vector v("D. Lee") = [0 0 1 3 4 0], provided that the dimensions are ordered by coauthors' last name and values in the vector represent the number of papers that the author represented by the dimension coauthored with "D. Lee". For example, the value, 1, in the third dimension represents the number of papers that "T. Han" coauthored with "D. Lee". Similarly, we have v("D. W. Lee") = [0 3 0 5 7 0] and v("D. Y. Lee") = [3 0 0 1 0 5]. In order to measure the similarity between the vectors, v and w, we use the simple cosine similarity, an angle between two vectors, defined as: $\cos\theta = \frac{c_c \cdot a_c}{\|c_c\| \cdot \|a_c\|}$. In the same procedure, we can measure Cosine similarity using titles and venues.

3.3 Analyzing Duplicate Entity Graph

To identify variant names and create the duplicate entity graph, a string similarity metric uses contex information based on the hypothesis that if two author names are identical, they will share more number of common tokens in their co-author names and titles/venues. If two nodes are connected to each other in the graph, they are likely to be name variants. By observing the topological features (i.e., degree, distance, betweenness, and clustering coefficient distribution) of the graph, we can be aware of name disambiguation problem more precisely. In other words, investigate if the graph is random, regular, scale-free, or small world network. Note that there exist on average 3 variant names in our data set that we manually collected from ACM digital library. Thus, we have our own point of view that if a string similarity method finds variant names well, the duplicate entity graph should show the topological features of random graphs. Otherwise, for instance, if it is a scale-free network, the used similarity metric and context information are not effective to identify redundant entities. In addition, we further examine that the duplicate entity graph is affected by different types of string similarity measures as well as by different attributes. The definitions of measuring topological features in the duplicate entity graph are as follows:

- **Degree**: The fraction of nodes of which each has $k_i = \frac{k_i}{N}$, where $1 \leq i \leq N$ and k_i is # of degrees of node i.
- **Distance** (GD): Given any pair of nodes in the duplicate entity graph, *geodesic* of the pair denotes the shortest path between them, and the geodesic distance is the minimum number of edges in the path.. Then, the average distance is the average of all pair-wise geodesic distances of nodes in a graph.
- **Betweenness**: The betweenness of a node v is measured by $\sum_{x,y,v \in G} \frac{d(x,y;v)}{d(x,y)}$, where $d(x,y)$ is a geodesic between x and y, and $d(x,y;v)$ is a geodesic between x and y passing through v. As the betweenness of a node v is high, v is considered as the central person in the community.
- **Clustering coefficient** (CC): Given a node v, let $\Gamma(v)$ be the neighbors of v, and $E(\Gamma(v))$ be the edges of nodes in $\Gamma(v)$. Then, the clustering coefficient of v is $\frac{2 \cdot |E(\Gamma(v))|}{(|\Gamma(v)|)(|(\Gamma(v))|-1)}$. In this case, the clustering coefficient of a graph G is the average clustering coefficient of all nodes in G, measuring how many edges actually occur, compared to the fully connected graph.

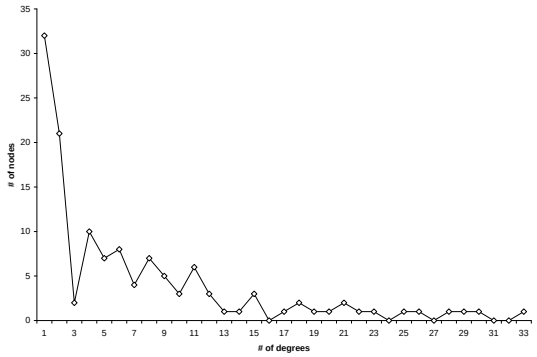


Figure 2. Digree distribution.

4 Experimental Validation

5 Set-Up

For our experiment, we used 128 "real" author names and their variants, manually collected from ACM Portal. For example, we manually verify that two author names "Chong Kwan Un", "C. K. Un", and "Chong K. Un" are the same author name in ACM Digital Library.

The three string similarity metrics were implemented by SecondString in http://secondstring.sourceforge.net/. For the input data of each similarity metric, we used co-author names, titles, and venue tokens as context information. That is, a set of co-author names of "Alon Y. Halevy", s set of title tokens of "Alon Y. Halevy", or a set of venue tokens of "Alon Y. Halevy". We empirically determined that the proper threshold value $\theta = 0.1, 0.4$, or 0.9 for Jaccard, Cosine similarity, and Edit distance, respectively.

5.1 Results

In our test data set, the number of author names is 43. Each author name corresponds to 2.98 name variants. For instance, an author name entity "Chun Wu Leng" and the

name variant "Chun-Wu Leng" exist in our data set. In the example, we call "Chun Wu Leng" a *representative* name while "Chun-Wu Leng" a *variant* name. To determine which one becomes the representative or variant name, we randomly select one of them as a representative name, and the rest as variant names. In case of "Levy", there are five split names. However, there are two duplicate names in most cases. Since a duplicate entity graph consists of 43 representative names and their variants, the total number of nodes is 128. Figure 5 shows the duplicate entity graph with 128 nodes, which is formed with the output data set generated by Cosine similarity metric, of which the context information is *co-authors* and $\theta = 0.4$. Each edge between two nodes i and j denotes that j is a variant name of i. When we observe the graph in Figure 5, we easily find duplicate entities. For instance, there is a subgraph of "Levy", such as "Alon Halevy", "Alon Levy", "Alon Y. Halevy", "A. Y. Halevy", and "Alon Y. Levy", at the bottom of Figure 5. As another example, take a look at nodes of "Laks V.S. Lakshmanan" and "V.S. Lakshmanan", where the two nodes are connected each other. In this case, using Cosine similarity metric, we identify variant names.

Now let us go back to our main point. Note that each representative name has at most 2 name variants in our data set. Therefore we argue that *if string similarity metrics, such as Jaccard, Cosine similarity, or Edit distance, identify name variants effectively, a duplicate entity graph should comprise a lot of forests and show topological properties of random graphs*. To verify our claim, Figure 2 depicts the degree distribution of the duplicate entity graph, which is formed with the output data set generated by Cosine similarity metric, of which the context information is *co-authors* and $\theta = 0.4$. Unexpectedly, the figure shows that the graph is a scale-free network. That is, only a few nodes have a considerable number of degrees which the most nodes have a small number of degrees. The distribution of numbers of degrees per node is power-low degree distribution, which is the main topological property of scale-free networks (e.g., WWW). As a result, a scale-free property of our duplicate entity graph implies that Cosine similarity metric does not work to find duplicate entities in our data set. This is why the graph contains false positive entities as well as variant names. As for an name entity, the corresponding false positive entity stands for a redundant entity determined by a certain string similarity metric but it is not true. According to our manual investigation, the most false positives are co-authors. For example, suppose that "J. D. Ullman" is a variant name of "Jeffrey Ullman" while "Alon Aho" is a co-author name entity (false positive) of "Jeffrey Ullman". Cosine similarity(context("Jeffrey Ullman"), context("J. D. Ullman"))=0.8. Cosine similarity(context("Jeffrey Ullman"), context("Alon Aho"))=0.8. If we assume that $\theta = 0.6$, "J. D. Ullman" and "Alon Aho" are redundant names of "Jeffrey Ullman". Thus the node "Alon Aho" is incident to "Jeffrey Ullman" in the duplicate entity graph. Consequently, we conclude that many false positives make a scale-free network and furthermore Cosine similarity metric does not find identical author names effectively.

Moreover, Figure 3 shows topological features of our duplicate entity graph. In other words, we measure average degree, distance distribution, betweenness distribution, and clustering coefficient distribution of three duplicate entity graphs formed by Jaccard, Cosine similarity, and Edit distance. Observing such various features, we become aware of the effectiveness of three different string similarity methods. In all figures in Figure 3, Jaccard and Cosine similarity show similar pattern of results. For instance, common geodesic distance of pairs of nodes in duplicate entity graphs created by Jaccard and Cosine similarity is 3. On the other hand, common geodesic distance in the duplicate entity graph formed by Edit distance is 2. This result indicates that Jaccard and Cosine similarity find redundant entities in the similar approach. Note that Jaccard works based on set operations of *tokens* regarding context information. Similarly, Cosine similarity is to first represent *tokens* of context information as vectors and take the close angle between vectors.

In particular, Edit distance is significantly different from the rest similarity metrics in the aspects of all features. For instance, in the graph of clustering coefficient distribution, clustering coefficient values of Edit distance are relatively higher than Jaccard and Cosine similarity. This observation indicates that many nodes in the duplicate entity graph in terms of Edit distance are more number of neighbors than Jaccard and Cosine similarity. That is, each node has many edges and is connected to false positives. Through this result, we can know that Edit distance is not effective method, compared to Jaccard and Cosine similarity. Please note that Edit distance measures # of character insertion, deletion, and substitution operations to transform context("Jeffrey Ullman") and context("J. D. Ullman"). In general, context information is the long length of string which consists of co-author name tokens, title tokens, or veneu tokens. As the length of context information is increased, the number of operations is increased as well. Hence Edit distance is the poorest method to find split name entities.

For each string similarity metric, we make use of three different context information. In other words, context("Jeffrey Ullman") can be his co-author names, title tokens or venue tokens of "Jeffrey Ullman" citations. To see the effectiveness of different context information, Figure 4 illustrates the four topological features of the duplicate entity graph in terms of Jaccard with $\theta = 0.4$. In all features, using co-authors and titles as context information is better than venues. For instance, using co-authors and titles

is smaller number of degrees than using venues. Through these results, co-authors and titles are worth as input data for similarity metrics. In many cases of venue information, the data is empty and shortened conference or journal names, say "VLDB" or "ICDE". This is why venue information is not good context information for string similarity metrics.

Table 2 summarizes features of the duplicate entity graph formed by Cosine similarity with $\theta = 0.4$.

6 Conclusion

In this paper, we project name disambiguation problem to graph representation, and then analyze graphs using social network analysis. Overall, according to our experimental results, the three string similarity metrics do not work effectively due to a scale-free topological feature. Furthermore, Jaccard and Cosine similarity using context information of co-authors and title tokens are the best to identify redundant entities.

As our extended future work, to determine proper threshold values for similarity metrics so to remove false positives, we can view that clustering coefficients are an effective method in our experiments. The clustering coefficent values are proportional to average recalls. If we set to θ a certain clustering coefficient value α, where there is a big gap between α and $\alpha + 1$. In this idea, we need to experiment intensively.

As our another work, we propose how to construct effective evaluation tools. To improve the quality of data (QoD), we need the approach to detect duplicate entities and then consolidate them. However, so far, none focuses on constructing a real evaluation tool for QoD. More specifically, our approach deals with the following two questions:

- Given a database, how can we know if it is really clean?
- Suppose that the database is so dirty that one of data cleaning methods is required to use. Then, how can we know if the method works without any solution set?

As our solution, we construct two graph models of dirty and clean data. Then, find the properties that clean data should have unlike dirty data. If such properties are observed in the given data (or the data after a cleaning method is performed), the data will be clean.

References

[1] R. Agrawal and R. Srikant. "Searching with Numbers ".

[2] M. Bilenko, R. Mooney, W. Cohen, P. Ravikumar, and S. Fienberg. "Adaptive Name-Matching in Information Integration". *IEEE Intelligent System*, 18(5):16–23, 2003.

[3] M. Bilenko, R.J. Mooney, W.W. Cohen, P. Ravikumar, and S.E. Fienberg. "Adaptive Name Matching in Information Integration". *IEEE Trans. on Computers*.

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

[5] W. A. Gale and K. W. Church. "A Program for Aligning Sentences in Bilingual Corpora". *Computational Linguistics*, 1993.

[6] L.E. Gill. "OX-LINK: The Oxford Medical Record Linkage System ".

[7] L. Gravano, P. G. Ipeirotis, N. Koudas, and D. Srivastava. "Text Joins in an RDBMS for Web Data Integration". In *Int'l World Wide Web Conf. (WWW)*, 2003.

[8] M. A. Hernandez and S. J. Stolfo. "The Merge/Purge Problem for Large Databases". In *ACM SIGMOD*, 1995.

[9] J. A. Hylton. *"Identifying and Merging Related Bibliographic Records"*. PhD thesis, Dept. of EECS, MIT, 1996. LCS Technical Report MIT/LCS/TR-678.

[10] M.A. Jaro. "Unimatch: A Record Linkage System: User's Manual ". Technical report, US Bureau of the Census, Washington, D.C., 1976.

[11] N. Koudas, A. Marathe, and D. Srivastava. "Flexible String Matching against Large Databases in Practice ".

[12] G.M. Landau and U. Vishkin. "Fast Parallel and Serial Approximate String Matching".

[13] V.I. Levenshtein. "Binary Codes Capable of Correcting Deletions, Insertions and Reversals".

[14] A.E. Monge and C.P. Elkan. "The Field Matching Problem: Alogirthms and Applications ".

[15] S.B. Needleman and C.D. Wunsch. "A General Method Applicalicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins".

[16] B.W. On, N. Koudas, D. Lee, and D. Srivastava. "Gropu Linkage ".

[17] L. Philips. "Hanging on the Metaphone ".

[18] L. Philips. "The Double Metaphone Search Algorithm ".

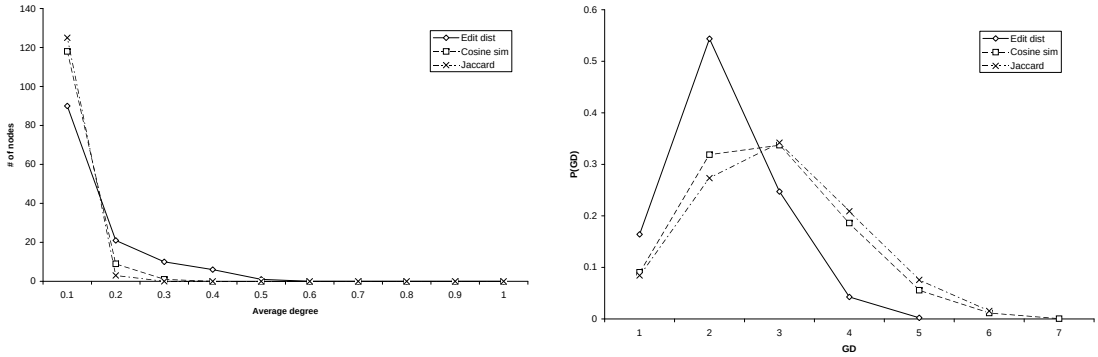
[19] E.S. Ristad and P.N. Yianilos. "Learning String Edit Distance". *IEEE Trans. on Computers*.

[20] R.C. Russell. "U.S. Patent 1,261,167 ".

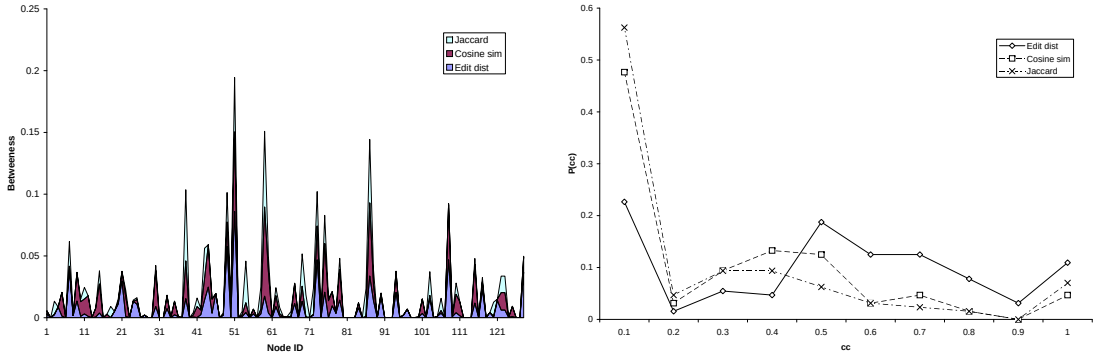
[21] R.C. Russell. "U.S. Patent 1,435,663 ".

[22] T.F. Smith and M.S. Waterman. "Identification of Common Molecular Subsequences".

[23] M.S. Waterman, T.F. Smith, and W.A. Beyer. "Some Biological Sequence Metrics".

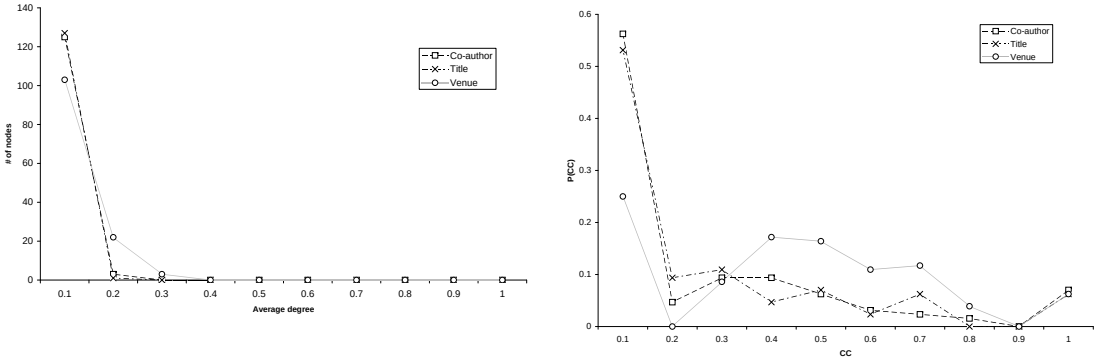


(a) Degree distribution (b) Distance distribution



(c) Betweenness distribution (d) Clustering coefficient distribution

Figure 3. Comparisons of Jaccard, Cosine similarity, and Edit distance.



(a) Average degree per node (b) Clustering coefficient distribution

Figure 4. Comparisons of co-authors, titles, and venues. Distance and betweenness results show similar patterns to degree and clustering coefficient distributions. Due to space limitation, we leave out the results.

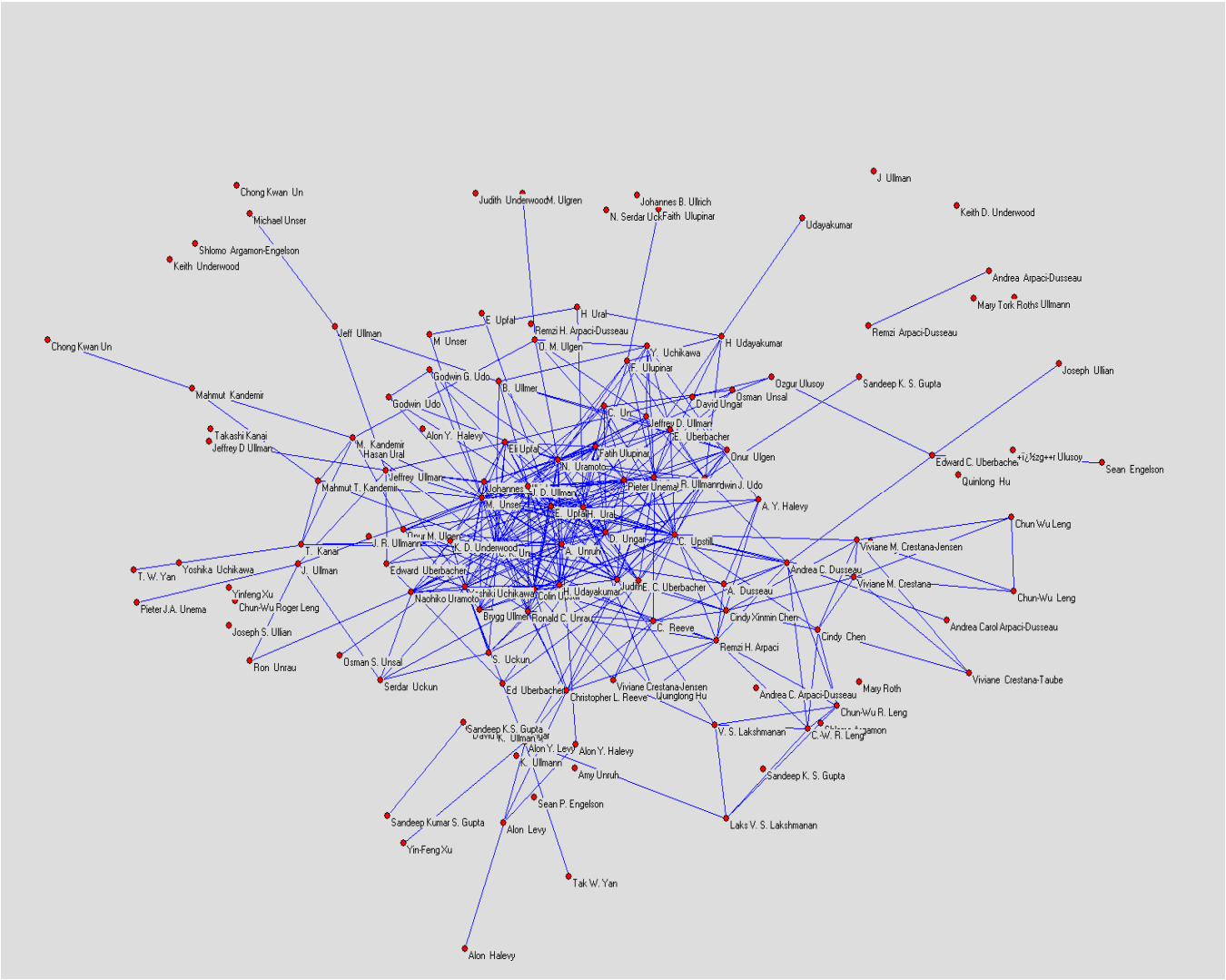


Figure 5. Screen-shot of a duplicate entity graph of using Pajek.

Network all degree centralization	0.17973
# of unreachable pairs	10058
Average distance among reachable pairs	3.02743
The most distance nodes	“Quinglong Hu” (23) and “Keith D. Underwood” (37)
Distance	7
Network betweenness centralization	0.10292

Table 2. Summary.