An Analysis of Named Entity Disambiguation in Social Networks

JADERICK P. PABICO
Institute of Computer Science, University of the Philippines Los Baños, Laguna
jppabico@uplb.edu.ph

Date Received: June 25, 2014; Date Published: August 15, 2014

Abstract – *The solution to the problem of disambiguating named entities in online social networks is introduced in this paper. The problem is solved via a two-step reduction process: A reduction to the Maximal Common Subgraph Problem (MaxSubgraph), and then a further reduction to the Maximal Clique Problem (MaxClique). An analysis of this process provides an upper bound of the runtime of the solution which was found to run in* $O(|V|^k k^2)$, *where* $|V|$ *is the maximum cardinality of vertices of the social networks and k is the size of the clique.*

***Keywords* –** name disambiguation, social networks, most common subgraph

I. INTRODUCTION

The advent of the so-called *Social Web* over the Internet has impacted the way people live in the digital age. The reason for this is that the social web has become a ubiquitous tool for people to meet, communicate, and collaborate with other people. Some examples of these social webs are the social networking sites Instagram [1] and Facebook [2], and the social media sites Digg [3] and Flickr [4]. Figure 1 shows one of the many visualizations of the inferred conceptual framework of the current social web as utilized by and in the point of view of an account owner. Because of the exponential proliferation of social web sites offering varied services and tools for meeting, communicating, and collaborating with other people, most social web users are tempted to create accounts in some of these sites. For example, a person P who created an account a_1 in Google+ [5] may create an account a_2 in Facebook, and another account a_3 in Twitter [6], often under different names or aliases. Generally, P may have n named entities $a_1, a_2, ..., a_n$ in n social networks $SN_1, SN_2, ..., SN_n$, respectively. Thus, in the context of social networking, named entity disambiguation is the problem of determining whether P, identified under the account a_1 in an online social network SN_1, is also the same person under account a_2 in another social network SN_2.

The *problem of disambiguating named entities* (NEDP) has already been defined and given solution in the field of information retrieval and data mining, particularly in advanced, large, and distributed databases.

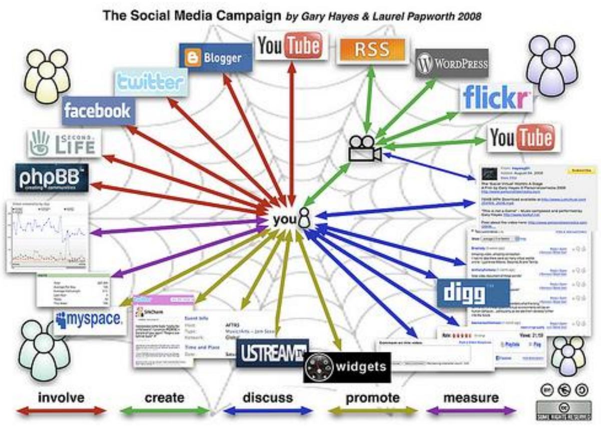

Figure 1. A popular visualization of the conceptual framework of the Social Web by Hayes [7] out of the many visualizations that exist.

When non-unique values such as names are used as the identifier of entities, confusion may occur due to the values' inherent homonyms, particularly when part of the names are used. For example, if only the first and last names are used as a combined identifier for a person, one cannot distinguish "Benigno Aquino, Jr.," the martyred senator, from "Benigno Aquino, III," the current President of the Philippines. Given a list of m documents $D = \{d_1, d_2, ..., d_m\}$ with all sharing an identical author name P but might actually be referring to different persons, the task is to assign each document

d_i to some cluster C_j, $1 \leq j \leq m$, where each cluster contains documents written by the same person. Theoretically, by viewing D as a vector of multiple dimensions and each $d_i \in D$ as a point in the multi-dimensional space, one can obtain a straightforward solution by using clustering methods to solve this problem. However, as speculated by Wang, *et al*. [8], such simple solution does not work well with this task because points in the multi-dimensional space is sparse (i.e., m is low). Thus, several approaches have been proposed for name disambiguation in various databases such as the scientific citation data [9–12], the WWW pages data [13, 14], the e-mail data [15], and the movie data [16]. Most of these approaches use machine learning and computational intelligence techniques which are suited for sparse, noisy, multi-modal, and multi-dimensional data, but however are space- and time-tractable approximation procedures that only find near-optimal solution sets. These solutions use co-references and co-occurrences of features of documents d_i and d_j to either learn or infer similarities between the named entities $P_i \in d_i$ and $P_j \in d_j$, $i \neq j$.

Finding similarities between two structures, wherein such structures are represented mathematically as graphs, may be solved by graph similarity (GS) techniques. GS has long been applied by various researchers to solve real-world problems. For example, in Chemistry, chemical compounds have a natural graph structure wherein vertices represent atoms and edges represent inter-atomic bonds. When experimental chemists develop a new compound, they compare its molecular structure to a database of compounds and look for those which are structurally identical [17–20]. In this kind of problem, the obvious computational solution is via GS. In biology, on the other hand, the mapping of metabolic networks into enzyme graphs has become a useful model for understanding the nature of life. In the enzyme graph, biological enzymes are represented as vertices while catalytic reactions are represented by edges. Biologists use these kind of mathematical model to explore different paths through the network in order to predict a behavior or explain some experimental data [20–26]. In computer vision, images that are being processed are represented as an attributed graph. Using GS techniques, an image processing procedure may be able to identify interesting similar features between two images [27], which later could be used in automated systems such as in optical character recognition or in biometric identification.

This paper presents an exploration of the utility of GS as an alternative solution to NEDP, particularly for named entities in social networks. By leveraging the inherent structure of the social network where the named entity belongs, various graph theoretic techniques may be used to augment the current machine learning and computational intelligence techniques. Given two graphs G_1 and G_2 that respectively represent the structure of the social networks SN_1 and SN_2, the NEDP is transformed to a similar problem of finding a common subgraph of G_1 and G_2 known in the graph theory discipline as the Maximum Common Subgraph Problem (MaxSubgraph). MaxSubgraph is solved by transforming it further to the problem of searching the largest clique of a compatibility graph associated with G_1 and G_2, otherwise known as the Maximum Clique Problem (MaxClique). The worst-case running time analysis of MaxClique provided a solution that runs no worse than $O(n^k k^2)$.

This paper is organized as follows: NEDP is first described as encountered in the area of large-scale databases and text mining in Section 2. It is argued that the most common methods may be improved if the topology of the social network where the named entity belongs is exploited to augment the disambiguation process. Then the paper proceeds to define the NEDP for named entities in social networks in Section 3. In Section 4, a worst-case analysis of the algorithm for solving MaxClique, which gives the upper bound for solving MaxGraph, is provided. Finally, conclusion is given in Section 5.

II. Named Entity Disambiguation Problem

In the area of database and text mining, a problem is encountered when records are identified with identifiers that do not uniquely tie with the entities. This problem is called the *record deduplication problem* (RDP) where one needs to decide whether two records in a database D refer to the same entity or not. RDP is a widespread problem in any large-scale database, specifically in the area of text mining where creation of records are done automatically. A special case of RDP is a problem called author disambiguation problem, or more generally named entity disambiguation problem (NEDP). NEDP is a widespread problem in digital publication libraries, such as CiteSeer, DBLP, Rexa, and Google Scholar. NEDP is a difficult problem to solve because of the inherent abbreviations, misspellings, and extraction errors brought about by automated text mining crawlers and web spiders. For example, the abbreviated name “B. Aquino” is a cause of ambiguity because one can not identify whether the name refers to “Benigno

Aquino, Jr.," "Benigno Aquino, III," or a less-popular "Boy Aquino." Also, one cannot be sure whether the name "Aqunio" is just a misspelled "Aquino," or there really exists such a less-known name as "Aqunio." Similarly, one cannot be sure either whether the name "Aquinas" was just a result of an extracting error by an optical character recognition software, or really such a name exists. Indeed, the problem becomes much more difficult if the existence of the other name does not provide any doubt to the existence of its owner (e.g., "Boy Aquino" is a name of a real person). In this kind of decision-making process, the "benefit of the doubt" is usually given to the less obvious names.

Most solutions to NEDP that are available in the literature [8, 28–33], specifically those that apply supervised machine learning techniques, have more or less the following general procedure:

1. Using a database of positive and negative examples of name duplicates, train a 0-1 classifier to differentiate whether a given pair of named entities are duplicates;
2. Apply the classifier to each pair of named entities perceived to be ambiguous; and,
3. Combine the classification predictions to cluster the records into duplicate sets.

These approaches are quite accurate with respect to most datasets in this problem domain, while they are generally attractive to researchers because they are built upon existing machine learning technologies. These technologies are that of the known classification and clustering techniques. However, because the core of these approaches are classifiers over record pairs namely that of the feature vector and the classification, most of these approaches fail to consider an important feature of the named entity itself, particularly, those entities who belong to various social networks. This important feature that one must consider is the topology of the social networks where the entity belongs.

III. SOCIAL NETWORKS AS GRAPHS

The social network *SN* is abstractly represented as a graph $G(V, E)$ composed of social members, such as humans, represented as a set V of n vertices $v_0, v_1, ..., v_{n-1}$. The relationship of a member v_i to any other member v_j, $\forall\, i \neq j$, is represented as an edge (i, j). The relationship of all members with the other members is a set E of edges, and can be mathematically represented as an $n \times n$ adjacency matrix M. The matrix element $M_{i,j} = 1$ if v_i is related in someway to v_j. Otherwise, $M_{i,j} = 0$. Note that the relationship is symmetric such that v_i having a relationship with v_j also means v_j having a relationship with v_i. Thus, $M_{i,j} = M_{j,i}$. Without loss of generality, the diagonal elements of the matrix $M_{i,i} = 0$, $\forall\, i$. For relationships whose degrees r can be quantified, the edge (i, j) is labeled as r, while $0 \leq M_{i,j} \leq 1$, where $M_{i,j} = r > 0$ if v_i is related to v_j to some degree, otherwise $Mi,j = 0$. In this case, the relationship is no longer symmetric. Under these degreed relationships, however, measuring r is quite difficult to achieve objectively for most real world *SN*'s. In this paper, it is assumed that $M_{i,j}$ is binary. The more general $0 \leq M_{i,j} \leq 1$ will be considered in the future.

The degree Δ_i of a vertex v_i counts the number of humans that member v_i has a relationship with. Thus, $\Delta_i = \sum_{j=1..n} M_{i,j}$. Recently, the frequency distribution $\rho(\Delta)$ of the degree in *SN* has been found by various researchers [34–36] to asymptotically follow the power law distribution of the form $\rho(\Delta) = \alpha \times \Delta^{\varphi}$. For social networks, and all other biological networks, the power usually takes the value $-3 \leq \varphi \leq -2$. Having $\rho(\Delta) \sim \alpha \times \Delta^{\varphi}$ makes *SN* scale-free [35].

Given a named entity P identified as an account holder a_1 in social network SN_1, and given another account a_2 in social network SN_2, the problem is to determine whether a_2 also belongs to P. Given the numerous existence of social networking sites nowadays, one can generally ask the question that given a list of account names $A = \{a_1, a_2, ..., a_n\}$ from respective social networks $SN_1, SN_2, ..., SN_n$, do the accounts in A belong to a single named entity P? The solution to this problem has many real-world applications. For example, in the area of automated social network aggregation, an aggregator can automatically suggest to a potential user which particular accounts from which particular social network may be aggregated. In the point of view of maintaining a social network site from duplicate accounts, two accounts from the same person may be easily deduplicated. Another very important application is in the area of law enforcement where a software can be used to automatically relate a suspect's known social network to his other unknown social networks in an effort to discover evidences. The topology of the social network of P can easily be measured because most social networking sites offer API to query its respective databases. Thus, if a graph G_1 represents the topology of the immediate neighbors of P in $SN1$, and another graph G_2 for SN_2, one can use GS measures to determine whether an account in SN_2 also refers to P.

IV. WORST CASE ANALYSIS

In this section, the *time complexity* of MaxGraph based on the *reduction* introduced by Kann [37], and later utilized by Raymond and Willett [18], is discussed. Time complexity is an analysis that classifies solutions to problems according to the solutions' inherrent difficulty in terms of the number of steps they will take to solve the problem. Reduction is the transformation of a problem to another problem whose time complexity is already known. The reduction of MaxGraph into MaxClique is likewise discussed here. For the sake of the readers who are not familiar with the terminologies, some preliminaries were discussed first such as defining the Vertex Cover Problem (VCP) and the Clique Problem (CP). For all the discussions, a graph $G(V, E)$ is considered.

Definition 1. Define $G[V']$ as the induced subgraph of G if it consists of $V' \subseteq V$ and those edges of G with both vertices in V'.

Definition 2. Two graphs $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ are isomorphic if there exists a *one-to-one correspondence* f: $V_1 \rightarrow V_2$ such that for every $(v_1)_1, (v_2)_1 \in V_1$, $\{(v_1)_1, (v_2)_1\} \in E_1$ if and only if $\{f((v_1)_1), f((v_2)_1)\} \in E_2$. The one-to-one correspondence f is also called a *bijection function* and is an *isomorphism*.

Definition 3. A *common subgraph* of G_1 and G_2 is a set of ordered pairs $\{((v_1)_1, (v_2)_1), ((v_1)_2, (v_2)_2), \ldots, ((v_1)_k, (v_2)_k)\}$ such that the function $f : \{(v_1)_1, (v_1)_2, \ldots, (v_1)_k\} \rightarrow \{(v_2)_1, (v_2)_2, \ldots, (v_2)_k\}$ defined by $f((v_1)_i) = (v_2)_i$ $(1 \leq i \leq k)$ is an isomorphism between $G_1[V_1']$ and $G_2[V_2']$ (note that $V_1' = \{(v_1)_1, (v_1)_2, \ldots, (v_1)_k\}$ and $V_2' = \{(v_2)_1, (v_2)_2, \ldots, (v_2)_k\}$).

Definition 4. VCP-k is a problem of finding if G has a vertex cover of k vertices.

Definition 5. CP-k is a problem of finding if G has a clique of k vertices.

The VCP-k requires one to find a subgraph in G, called a *cover* C, with k vertices such that every edge in G has at least one end in C. In Figure 2a for example, the VCP-3 of G is $C = (\{v_1, v_2, v_6\}, \{(v_1, v_2), (v_2, v_6)\})$ because every edge $E = \{(v_1, v_4), (v_1, v_2), (v_2, v_4), (v_2, v_5), (v_2, v_3), (v_2, v_6)\}$ has at least one end in the vertex set of C. Given a graph G with $n = |V|$ vertices, it has been shown by Chen, *et al*. [38] that the algorithm for solving the VCP-k runs in time $O(kn + 1.286^k)$.

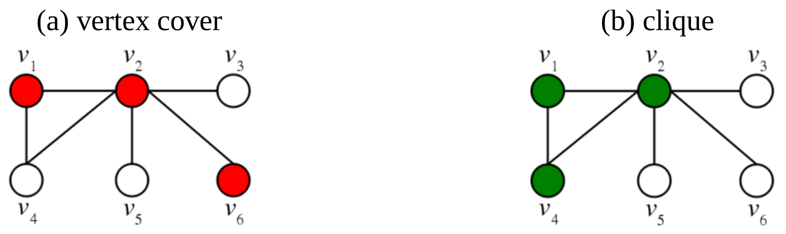

Figure 2. Given an example graph $G = (\{v_1, v_2, \ldots, v_6\}, \{(v_1, v_4), (v_1, v_2), (v_2, v_4), (v_2, v_5), (v_2, v_3), (v_2, v_6)\})$, then (a) a *vertex cover* C of G with $k = 3$ are the red-colored vertices, while (b) a *clique* of G with $k = 3$ are the green-colored vertices.

The CP-k, on the other hand, requires one to find a subset of k vertices in G such that there is an edge in G between any two of these k vertices. In other words, CP is finding a set of k vertices that induce a complete subgraph of G (Figure 2b). Based on the enumeration of all the vertex subsets of size k, the running time of an algorithm that solves CP must not be worse than $O(n^k)$.

Given two graphs $G_1(U, E_1)$ and $G_2(V, E_2)$, a new graph $G_3(W, E_3)$ is derived using the following steps. Let $W = U \times V$ be a set of vertex pairs. Call any two pairs $<u_1, v_1>$ and $<u_2, v_2>$ as *compatible* if $(u_1,u_2) \in U$ and $(v_1,v_2) \in V$ (i.e., if they preserve the edge relation) or $(u_1,u_2) \in U$ and $(v_1,v_2) \in V$. Preserving the edge relations means that there is an edge between u_1 and u_2 if and only if there is an edge between v_1 and v_2. Then let E_3 be the set of compatible edges (Figure 3a). A k-clique in G_3 is a matching between two induced subgraphs $G_1[U']$ and $G_2[V']$, each of size k. The two subgraphs are isomorphic since the compatible pairs preserve the edge relations. The new graph G_3 is called the *modular product* of G_1 and G_2. *The induced subgraph in G_3 provides all possible solutions to the NEDP*.

Without loss of generality, it is assumed that $n = |V_1| = |V_2|$. When G_3 is constructed, the cardinality of V_3 is constrained to n^2. By observing the structure of G_3, one can infer that G_3 is an n-partite graph (Figure 3b). Here, the vertices are partitioned in n disjoint partitions, with each partition having n vertices. Thus, one can use a matrix M to denote the n^2 pairs of vertices of the n-partite graph as follows: The n pairs of vertices of the first row $<(\mathrm{v}_1)_1, (v_2)_i>$, $\forall\, 1 \leq i \leq n$, belong to partition one of the n-partite graph. Likewise, the n pairs of vertices of the second row $<(v_1)_2, (v_2)_i>$, $\forall\, 1 \leq i \leq n$, belong to partition two of the n-partite graph. Thus, in general, the n pairs of vertices of the jth row $<(v_1)_j, (v_2)_i>$, $\forall\, 1 \leq i \leq n$, belong to partition j of the n-partite graph, $\forall\, 1 \leq j \leq n$. Notice here that there is no edge between any two vertices within the same partition. Edges are only present between two vertices that belong to two different partitions. Thus, at most, one vertex from each partition (among the n vertices) could be in a clique of the graph. To find CP-k via exhaustive enumeration, there will be n^k possible ways. Each possible way needs $O(k^2)$ time to check if it constructs a k-clique. Thus, this gives an algorithm a time complexity of $O(n^k k^2)$ for MaxGraph.

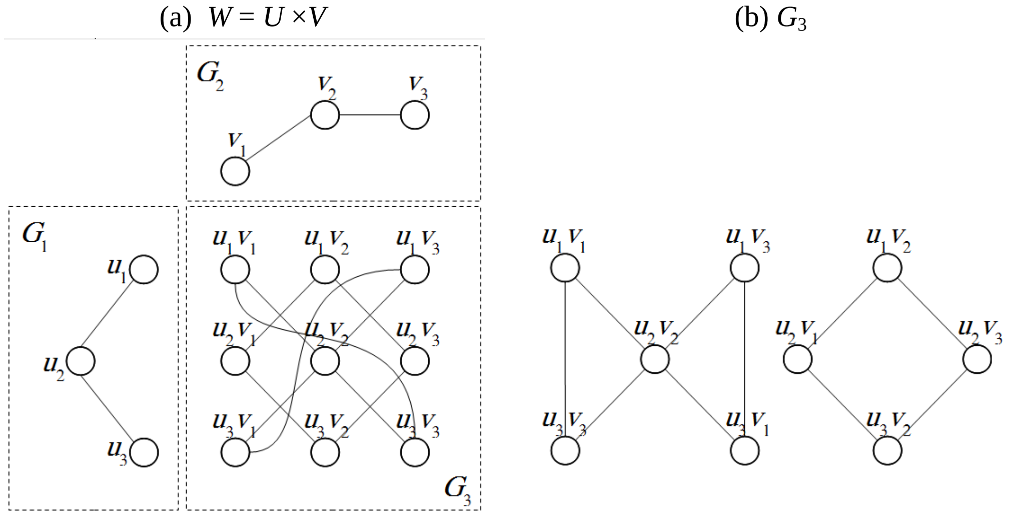

Figure 3. (a) An example modular product of $G_1(U,E_1)$ and $G_2(V,E_2)$ resulting to $G_3(W,E_3)$, where $U = \{u_1, u_2, u_3\}$, $V = \{v_1, v_2, v_3\}$ and $W = U \times V$. Notice that G_3 is a row-based and column-based 3-partite graph. (b) The resulting G_3 is a disconnected graph with two components. The first component on the left has two cliques: $\{u_1v_1, u_2v_2, u_3v_3\}$ and $\{u_1v_3, u_2v_2, u_3v_1\}$. The first clique means that the named entity u_1 is a match for the named entity v_1, u_2 for v_2, and u_3 for v_3.

A. Case when $k = n$

The case when the maximum clique size k is equal to n is studied in detail by Sze, *et al*. [39] where a fast and yet exact divide-and-conquer approach to solving CP-n was developed. Their idea is to subdivide the n-partite graph into several n_0-partite subgraphs, $n_0 < n$, and solve each smaller subgraphs independently using a branch-and-bound approach. The approach works perfectly well as long as the number of cliques n_0 in each subproblem is not too high. The worst case, however, has the same bound as above.

A. Case when $k = n - 1$

The analysis for $k = n - 1$ was discussed in detail by Sze, *et al*. [39]. Suppose the n-partite graph has a clique of size $k - 1$. One phantom vertex is added to each of the n partitions, and then edges from the phantom vertex to any non-phantom vertices are that are in the different partition are added. Here, a new graph G_3 which is an n-partite graph with $n + 1$ vertices in each partition is obtained. Graph G_3 has a clique of size n if and only if the original n-partite graph has a clique of size $n - 1$. The vertices of this new clique include the vertices of the original clique plus one phantom vertex. The same procedure discussed above will need a time $O((n + 1)^n n^2)$ to find the clique for G_3. After the clique is found, the phantom vertices is removed from the clique.

A. Case when $k = n – c$

When $k = n - c$, where integer $c > 1$, similar analysis as above is used. Here, c phantom vertices are added in each partition and these vertices are connected to any non-phantom ones in different partitions. The worst case running time when applying this algorithm is $O((n + c)^n n^2)$.

V. SUMMARY AND CONCLUSION

In this paper, the problem of named entity disambiguation in social networks is introduced. It is argued that the most common solution to name disambiguation problem using machine learning and computational intelligence techniques may be improved by exploiting the inherent topology of the social network where the named entity belongs. It is said that the immediate neighbors of a named entity in the social network may be considered as a graph, and similarly the immediate neighbors of an ambiguous name from another social network may also be considered as a graph. Thus, a graph similarity measure may be used to augment the current disambiguation process. The MaxGraph problem was used to describe the NEDP in social networks. MaxGraph was solved by a reduction to MaxClique. Then an analysis of the upper bound of the procedure was provided and found that the running time must not be worse than $O(|V|^k k^2)$, where $|V|$ is the maximum cardinality of vertices of the social networks and k is the size of the clique.

ACKNOWLEDGEMENTS

This research effort is funded by the Institute of Computer Science (ICS) of the University of the Philippines Los Baños (UPLB) under the research programs *Structural Characterization and Temporal Dynamics of Various Natural, Social and Artificial Networks in the Philippines* and *Evaluating Approaches for Modeling and Simulating the Structure and Dynamics of Artificial Societies*.

Ms. Fatima M. Moncada, University Research Associate of the PCAARRD-funded R&D Program *National Pork Traceability System*, is credited for her valuable suggestions in making this paper readable.

REFERENCES

[1] K. Systrom and M. Krieger. 2014. **Instagram**. Facebook, Inc. (http://www.instagram.com).

[2] M. Zuckerberg, E. Saverin, A. McCollum, D. Moskovitz, and C. Hughes. 2014. **Facebook**. Facebook, Inc. (http://www.facebook.com).

[3] J. Adelson and K. Rose. 2014. **Digg**. Digg, Inc. (http://www.digg.com).

[4] S. Butterfield and C. Fake. 2014. **Flickr**. Yahoo! Southeast Asia Pte. Ltd. (http://www.flickr.com).

[5] Google, Inc., 2014. **Google Plus** (http://plus.google.com).

[6] J. Dorsey, N. Glass, E. Williams and B. Stone. 2014. **Twitter**. Twitter, Inc. (http://www.twitter.com).

[7] G. Hayes. 2008. **The social media campaign** (http://www.personalizemedia.com/).

[8] F. Wang, J. Li, J. Tang, J. Zhang, and K. Wang. 2008. *Name disambiguation using atomic clusters*. In **The 9th International Conference on Web-Age Information Management (WAIM 2008)**, Zhangjiajie, China, pp 357–364 (DOI: 10.1109/WAIM.2008.96).

[9] H. Han, C.L. Giles, H. Zha, C. Li, and K. Tsioutsiouliklis. 2004. *Two supervised learning approaches for name disambiguation in author citations*. In **Proceedings of the ACM/IEEE Joint Conference on Digital Libraries (JCDL 2004)**, Tuczon, Arizona, USA, pp 296–305 (DOI: 10.1145/996350.996419).

[10] H. Han, H. Zha, and C.L. Giles. 2005. *Name disambiguation in author citations using a k-way spectral clustering method*. In **Proceedings of the 5th ACM/IEEE Joint Conference on Digital Libraries (JCDL 2005)**, Denver, Colorado, USA, pp 334–343 (DOI: 10.1145/1065385.1065462).

[11] H. Han, W. Xu, H. Zha, and C.L. Giles. 2005. *A hierarchical naive bayes mixture model for name disambiguation in author citations*. In **Proceedings of the 20th Annual ACM Symposium on Applied Computing (SAC 2005)**, Santa Fe, New Mexico, USA, pps 1065–1069 (DOI: 10.1145/1066677. 1066920).

[12] X. Yin, J. Han, and P.S. Yu. 2007. *Object distinction: Distinguishing object with identical names by link analysis*. In **Proceedings of the 23rd International Conference on Data Engineering (ICDE 2007)**, Istanbul, Turkey, pp 1242–1246.

[13] R. Bekkerman and A. McCallum. 2005. *Disambiguating web appearances of people in a social network*. In **Proceedings of the 14th International Conference on World Wide Web (WWW 2005)**, Chiba, Japan, pp 463–470 (DOI: 10.1145/1060745.1060813).

[14] G.S. Mann and D. Yarowsky. 2003. *Unsupervised personal name disambiguation*. In **Proceedings of the 7th Conference on Computational Natural Language Learning** (Human Language Technology-North American Chapter of the Association for Computational Linguistics), Edmonton, Canada, pp 33–40 (DOI: 10.3115/1119176.1119181).

[15] E. Minkov, W.W. Cohen, and A.Y. Ng. 2006. *Contextual search and name disambiguation in email using graphs*. In **Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval**, Seattle, Washington, USA, pp 27–34 (DOI: 10.1145/1148170.1148179).

[16] B. Malin, E. Airoldi, and K.M. Carley. 2005. *A network analysis model for disambiguation of names in lists*. **Computational and Mathematical Organization Theory** 11(2):119–139 (DOI: 10.1007/s10588-005-3940-3).

[17] A.T. Balaban. 1985. *Applications of graph theory in chemistry*. **Journal of Chemical Information and Computer Sciences** 25(3):334–343 (DOI: 10.1021/ci00047a033).

[18] J.W. Raymond and P. Willett. 2002. *Maximum common subgraph isomorphism algorithms for the matching of chemical structures*. **Journal of Computer-Aided Molecular Design** 16:521–533 (DOI: 10.1023/A:1021271615909).

[19] J.W. Raymond, E.J. Gardiner, and P. Willett. 2002. *Heuristics for similarity searching of chemical graphs using a maximum common edge subgraph algorithm*. **Journal of Chemical Information and Computer Sciences** 42(2):305–316 (DOI: 10.1021/ci010381f).

[20] M. Kanehisa and P. Bork. 2003. *Bioinformatics in the post-sequence era*. **Nature Genetics** 33:305–310 (DOI: 10.1038/ng1109).

[21] I. Koch, T. Lengauer, and E. Wanke. 1996. *An algorithm for finding common subtopologies in a set of protein structures*. **Journal of Computational Biology** 3(2): 289–306 (PMID: 8811488).

[22] I. Koch and T. Lengauer. 1997. *Detection of distant structural similarities in a set of proteins using a fast graph-based method*. In T. Gaasterland, P.D. Karp, K. Karplus, C. Ouzounis, C.s Sander, and A. Valencia (editors) **Proceedings of the 5th International Conference on Intelligent Systems for Molecular Biology**, Menlo Park, AAAI Press, pp 167–178.

[23] C.H. Schilling, D. Letscher, and B.O. Palsson. 2000. *Theory for the systematic definition of metabolic pathways and their use in interpreting metabolic function from a pathway-oriented perspective*. **Journal of Theoretical Biology** 203(3): 229–248 (PMID: 10716907).

[24] F. Chung, L.Y. Lu, T.G. Dewey, and D.J. Galas. 2003. *Duplication models for biological networks*. **Journal of Computational Biology** 10(5):677–687 (DOI: 10.1089/106652703322539024).

[25] M. Heymans and A.K. Singh. 2003. *Deriving phylogenetic trees from the similarity analysis of metabolic pathways*. **Bioinformatics** 19(1):i138–i146 (DOI: 10.1093/bioinformatics/btg1018).

[26] N.D. Price, J.L. Reed, J.A. Papin, I. Famili, and B.O. Palsson. 2003. *Analysis of metabolic capabilities using singular value decomposition of extreme pathway matrices*. **Biophysical Journal** 84(2):794–804 (PMID: 12547764).

[27] D. Conte, P. Foggia, C. Sansone, and M. Vento. 2004. *Thirty years of graph matching in pattern recognition*. **International Journal of Pattern Recognition and Artificial Intelligence** 18(3):265–298 (DOI: 10.1142/S0218001404003228).

[28] J. Huang, S. Ertekin, and C.L. Giles. 2006. *Efficient name disambiguation for large- scale databases*. In J. Fumkranz, T. Scheffer, and M. Spiliopoulou (editors) **Proceedings of the 10th European Conference in Principles and Practice of Knowledge Discovery in Databases (PKDD 2006)**, Berlin, Germany (Lecture Notes in Computer Science, volume 4213) pp 536–544. (DOI: 10.1007/11871637_53)

[29] S. Cucerzan. 2007. *Large-scale named entity disambiguation based on Wikipedia data*. In **Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning**, Prague, Czech Republic, Association for Computational Linguistics, pp 708–716.

[30] A. Culotta, P. Kanani, R. Hall, M. Wick, and A. Mccallum. 2007. *Author disambiguation using error-driven machine learning with a ranking loss function*. In **Proceedings of the 6th International Workshop on Information Integration on the Web (IIWeb 2007)**, Vancouver, Canada.

[31] V.I. Torvik and N.R. Smalheiser. 2009. *Author name disambiguation in MEDLINE*. **ACM Transactions on Knowledge Discovery from Data** 3(3):11(1)–11(29) (DOI: 10.1145/1552303.1552304).

[32] P.P. Kuksa and Y. Qi. 2010. *Semi-supervised bio-named entity recognition with word-codebook learning*. In **Proceedings of the 2010 SIAM International Conference on Data Mining (SDM 2010)**, Columbus, Ohio, USA, pp 25–36.

[33] F. Wang, J. Tang, J.i Li, and K. Wang. 2010. *A constraint-based topic modeling approach for name disambiguation*. **Frontiers of Computer Science in China** 4(1):100–111 (DOI: 10.1007/s11704-009-0064-9).

[34] A.L. Barabasi and R. Albert. 1999. *Emergence of scaling in random networks*. **Science** 286(5439):509–512 (DOI: 10.1126/science.286.5439.509).

[35] R. Albert and A.L. Barabasi. 2002. *Statistical mechanics of complex networks*. **Reviews of Modern Physics** 74:47–97 (DOI: 10.1103/RevModPhys.74.47).

[36] A.L. Barabasi and E. Bonabeau. 2003. *Scale-free networks*. **Scientific American** 288(5):50–59 (DOI: 10.1038/scientificamerican0503-60).

[37] V. Kann. 1992. *On the approximability of the maximum common subgraph problem*. In **Proceedings of the 9th Annual Symposium on Theoretical Aspects of Computer Science, Lecture Notes in Computer Science**, volume 577, Springer-Verlag, Cachan, France, pp 377–388 (DOI: 10.1007/3-540-55210-3_198).

[38] J. Chen, I.A. Kanj, and W. Jia. 2001. *Vertex cover: Further observations and further improvements*. **Journal of Algorithms** 41(2):280–301 (DOI: 10.1006/jagm.2001.1186).

[39] S.H. Sze, S. Lu, and J. Chen. 2004. *Integrating sample-driven and pattern- driven approaches in motif finding*. In **Proceedings of the 4th International Workshop on Algorithms in Bioinformatics (WABI 2004)**, Bergen, Norway, pp 438–449 (DOI: 10.1007/978-3-540-30219-3_37).