Intelligent Systems Conference 2017
7-8 September 2017 | London, UK

LUCID: Author Name Disambiguation using Graph Structural Clustering

Ijaz Hussain
Department of Computer Science
COMSATS Institute of Information Technology
Islamabad, Pakistan 78000

Sohail Asghar
Department of Computer Science
COMSATS Institute of Information Technology
Islamabad, Pakistan 78000

***Abstract*—Author name ambiguity may occur in two situations when multiple authors have the same name or the same author writes her name in multiple ways. The former is called homonym and the later is called synonym. Disambiguation of these ambiguous authors is a non-trivial job because there is a limited amount of information available in citations data set. In this paper, a graph structural clustering algorithm "LUCID: Author Name Disambiguation using Graph Structural Clustering" is proposed which disambiguates authors by using community detection algorithm and graph operations. In the first phase, LUCID performs some preprocessing tasks on data set and creates blocks of ambiguous authors. In the second phase co-authors graph is built and "SCAN: A Structural Clustering Algorithm for Networks" is applied to detect hubs, outliers, and clusters of nodes (author communities). The hub node that intersects with many clusters is considered as a homonym and resolved by splitting across this node. Finally, the synonyms are disambiguated using proposed hybrid similarity function. LUCID performance is evaluated using a real data set of Arnetminer. Results show that LUCID performance is overall better than baseline methods and it achieves 97% in terms of pairwise precision, 74% in pairwise recall and 82% in pairwise F1.**

***Keywords*—Author name disambiguation; structural clustering; co-authors graph; homonym resolver; synonyms resolver**

I. Introduction

Currently, Digital Libraries (DLs) such as DBLP[1], MEDLINE[2], Cite Seer[3] arXiv[4], MAS[5], Google Scholar[6], and BDBComp[7] are a big source of scholarly information retrieval. However, the majority of DLs are unable to correctly identify the oeuvre of an author due to author name ambiguity problem. This may happen in two different ways when multiple authors have the same name or the same author writes his name in multiple ways. The former is called homonym and the later is called synonym. For instance, when we searched DBLP (Digital Bibliography & Library Project) a name "Ajay Gupta", it returned multiple authors with exactly the same name "Ajay Gupta". Similarly, when we searched a name "A. Gupta" it returned greater than two hundred likely matches. Some bad effects of the author name ambiguity are wrong search results and incorrect attributions. Systems which resolve these ambiguities are known as Author Name Disambiguation (AND) systems [5, 18].

Several AND systems were presented in the last few years that used many different techniques from machine learning to data mining [1, 2, 3, 4, 6, 10, 13, 15, 16, 18, 19, 22]. However, some of them only resolved the homonyms [4, 24], some concentrated only on synonyms [4, 15] and a few required some auxiliary information like affiliation, email or related web page, or needed some hidden information such as the unknown number of ambiguous authors a priori or required some experts knowledge to estimate unknown parameters and thresholds [6, 11, 16, 17, 18]. Still, some others heavily relied on training data to train the proposed model [22, 23].

In this paper "LUCID: Author Name Disambiguation using Graph structural Clustering" is proposed to resolve author name ambiguity problems. LUCID starts working by preprocessing the data set and then forms ambiguous author blocks [6, 18], co-author's graph is built from the preprocessed data set. In the next step, "SCAN: A Structural Clustering Algorithm for Networks" is applied to the co-author's graph to find the hubs, outliers, and clusters of nodes (author communities) [27]. Hub nodes are considered as the homonyms and if multiple author communities are identified in the co-author's graph then these are considered as different homonyms and resolved by splitting these communities across the hub nodes. Synonyms are first identified and then resolved with the help of proposed hybrid similarity. In this similarity, both compared names should be compatible and their syntax similarity should also be greater than a defined threshold. Then the geodesic distance between these name nodes is found. If any two vertices fulfill the both conditions of hybrid similarity then these vertices are fused into one node. Co-authors attribute has been considered more discriminative feature than other citation features in many earlier studies [4, 18, 19, 21]. LUCID simultaneously solves homonyms as well as synonyms. One of the prominent features of LUCID is that it only required co-authors attribute to fully resolve the author name ambiguity problems.

LUCID performance is evaluated by comparing it with two state-of-the-art unsupervised name disambiguation methods HHC and GFAD [2, 18] using a data set of Arnetminer [20]. Despite only using co-authors attribute, LUCID performance is overall better than baseline methods from the perspective of representative clustering evaluation metrics. In summary, the main contributions of this work are:

[1] http://dblp.uni-trier.de
[2] http://www.medline.com
[3] http://citeseerx.ist.psu.edu
[4] http://arxiv.org
[5] http://academic.research.microsoft.com
[6] http://scholar.google.com.pk
[7] http://www.lbd.dcc.ufmg.br/bdbcomp

978-1-5090-6435-9/17/$31.00 ©2017 IEEE

- A homonym resolver algorithm is proposed that uses graph-based community detection algorithm to detect homonyms and graph operations to resolve them.
- A synonym resolver algorithm is also presented that uses proposed hybrid similarity to detect and resolve synonyms.
- Experiments on Arnetminer data set are performed to validate the performance of LUCID.

Rest of paper is structured as follows. In "Related Work" some related works to LUCID are discussed and their capabilities and limitations are highlighted. "Proposed Method: LUCID" describes in detail the working of LUCID. We simulated proposed algorithm on Arnetminer and compare it to baseline methods in "Experiments and Discussions". In the end, we conclude with a discussion of conclusions and future work in "Conclusions".

II. Related Work

Some AND methods used supervised techniques [7, 21, 23], other used unsupervised techniques [2, 16, 19, 26], still other used semi-supervised techniques [11, 28], and graph oriented techniques [4, 12, 18].

A boosted tree method was proposed in [23] for author name disambiguation. In the first step, they filtered authors by name and affiliation matches. Then similarity scores for publication attributes are computed and author screening is performed. Finally, boosted tree classifier was applied to the manually crafted data set. In [7] two extreme learning machine based algorithms were proposed for author name disambiguation. But this method requires training of a new classifier for each ambiguous author name. Similarly, in [21] a deep neural network approach was presented to automatically learn feature vectors and disambiguate those authors. They tested their proposed solution with Vietnameses data set, while they claimed that their proposed method can equally useful for any data set. They utilized the multi-column deep neural networks to improve the generalization capabilities of the system that is very similar to ensemble method bagging.

A simple two-step algorithm for AND using heuristic-based hierarchical clustering method was proposed by [2]. They resolved both homonyms and synonyms by exploiting similarities in publication attributes combined with some other heuristics. Initially, they clustered citations on the basis of some common co-author names. They created fragmented but very pure clusters. These clusters were then pair-wise compared and merged if they have a similarity greater than some predefined threshold in their title and venue. This process was iterative and successive iterations had a synergetic effect on disambiguation of authors, as more titles and venues were combined in these merged clusters. This process finished when no more fusions between clusters were possible or a number of specified iterations was reached. This method may produce different clusters for each run. A discrimination function in [16] was proposed that predicted true authors and false authors using logistic regression on Web of Science data set. They extracted true author papers from a large amount of retrieved papers by using two stage filtering. This method is not good for papers in which subject field and affiliation address do not vary too much. Wu et al. in [26] proposed an algorithm that used Dempster-Shafer Theory (DST) in conjunction with Shannon Entropy (SE) for author disambiguation. In the first step, some high-level features and their co-relation similarities were calculated. In next step, these features were fused using DST and SE. In clustering stage, they used three different convergence conditions for clustering algorithm namely–the preset number of clusters, the number of available evidence and distance between clusters.

A unified probabilistic framework to address the homonyms in DLs was proposed by [19]. They formalized the disambiguation problem using Markov random fields, in which the data were cohesive on both local attributes and relationships. They proposed algorithms that automatically estimate the number of unknown ambiguous authors and the required unknown parameters. A hybrid name disambiguation framework was presented in [28] by zhu et al. which not only used the traditional information (co-authors) but also web page genre information. This framework consisted of two main steps; web page genre identification and re-clustering model. A semi-supervised approach was proposed in [11], which offered an end to end solution for AND and it can also be used as a wrapper service for DLs. They used selected features of an author and citation record to disambiguate authors using unsupervised techniques. However, at some stages, they interacted with users to further purify the retrieved clusters.

"GHOST: GrapHical framewOrk for name diSambiguaTion" was presented in [4] that first modeled the relationships among publications using undirected graphs. They solved homonyms problem by iteratively finding valid paths, computing similarities, clustering with the help of affinity propagation algorithm and in the last using user feedback as a complementary tool to enhance the performance. This approach does not handle outliers and fails in the case of solo authors. A graph framework for author name disambiguation "GFAD" was proposed in [18] which exploited co-authors and citation titles. They modeled an undirected graph by denoting an author to a vertex and a co-author relation was indicated by an edge. Namesake problem was resolved by splitting the vertex that was common to multiple non-overlapping cycles so that each cycle correspond to a unique author. The heteronymous name problem was handled by merging multiple author vertices into one by identifying those vertices that actually represented a single author with different names. This method is computationally too expensive as it used Johnson's cycle detection algorithm. It removed outliers by comparing similarity among outliers with the help of cosine similarity measure. LUCID is similar to existing AND systems as it falls in the unsupervised category [2, 3, 4, 18]. But different from other AND systems [1, 11, 14], as it uses the graph structural clustering and graph operations. Also in it, a hybrid similarity is proposed that uses syntax similarity as well as graph geodesics to disambiguate synonyms.

III. Proposed Method: LUCID

We give some definitions for understanding the structural clustering specifically related to SCAN algorithm that are being utilized in LUCID. Then, LUCID is presented in subsequent paragraphs, whereas its architecture is shown in Fig. 1.

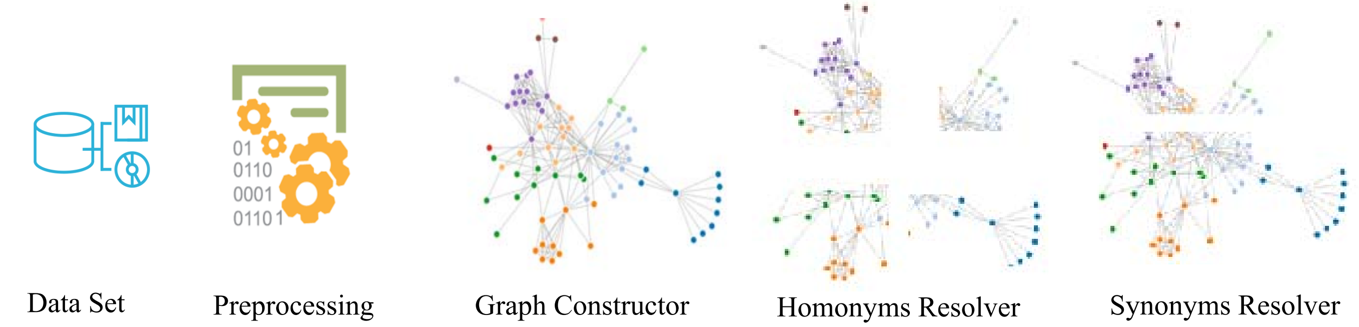


Fig. 1. Architecture of LUCID.

A. *Preliminaries*

We give these definitions here as used in [27] for the reader's convenience. Let $G = \{N, L\}$ be a graph, where N is a set of nodes and L is set of links in this graph G.

Definition 1. CORE

Core nodes have a minimum of μ neighbors with a structural similarity that is greater than the specified threshold ϵ. Clusters are grown from core nodes. Parameters μ and ϵ are the determinants of the clustering. Let $\epsilon \in \mathbb{R}$ and $\mu \in \mathbb{N}$. A node $n \in N$ is called a core according to ϵ and μ, if its ϵ-neighborhood contains at least μ nodes:

$$CORE_{\epsilon,\mu}(m) \Leftrightarrow \|N_\epsilon(m)\| \geq \mu \tag{1}$$

Definition 2. CLUSTERS

Now we can define a clustering of a network G w.r.t. the given parameters ϵ and μ as all structure-connected clusters in G. Let $\epsilon \in \mathbb{R}$ and $\mu \in \mathbb{N}$. A clustering P of network $G =< M, L >$ w.r.t. ϵ and μ consists of all structure-connected clusters w.r.t. ϵ and μ in G, formally:

$$CLUSTERS_{\epsilon,\mu}(P) \Leftrightarrow P = \{C \in N \| CLUSTER_{\epsilon,\mu}(C)\} \tag{2}$$

A node is either a member of a cluster, or it is isolated. If a node is not a member of any structure-connected clusters, it is either a hub or an outlier, depending on its neighborhood.

Definition 3. HUB

Let $\epsilon \in \mathbb{R}$ and $\mu \in \mathbb{N}$. For a given clustering P, i.e. $CLUSTERS_{\epsilon,\mu}(P)$, if an isolated node $n \in N$ has neighbors belonging to two or more different clusters w.r.t. ϵ and μ, it is a hub w.r.t. ϵ and μ, and formally,

$$HUB_{\epsilon,\mu}(N) \Leftrightarrow \forall(C) \in P \ : n \notin C \, \exists\, (p,q) \in \Gamma(n) : \exists(X,Y) \in P : X \neq Y : \wedge p \in X \wedge q \in Y \tag{3}$$

Definition 4. OUTLIER

For a given cluster P, i.e. $CLUSTERS_{\epsilon,\mu}(P)$, an isolated node $n \in N$ is an outlier if and only if all its neighbors either belong to only one cluster or do not belong to any cluster.

$$OUTLIER_{\epsilon,\mu}(N) \Leftrightarrow \forall(C) \in P \ : n \notin C \; \neg\exists\, (p,q) \in \Gamma(v) : \exists(X,Y) \in P : X \neq Y : \wedge p \in X \wedge q \in Y \tag{4}$$

We implemented SCAN algorithm as proposed by [27] in Python to detect hubs, outliers and clusters (author communities) from the co-author's graph. This step is given in LUCID algorithm at line 4. Interested readers are requested to read [27] for further details.

B. *Preprocessing and Graph Building Stage*

All the citations data set is extracted and pre-processed into different tokens, like authors, titles, and venues, etc. In blocking stage, similar names are grouped together if their similarity is higher than some predefined threshold value. We used fragment comparison method as it was proved to be effective for name disambiguation in many previous studies [2, 6]. Blocking stage is important as it affects the computations in the later stages of the name disambiguation algorithms. The Blocking stage returns 'l' blocks if given 'm' authors, it is shown in Fig. 2 .

The computation complexity is

$$\mathcal{O}(m^2) \tag{5}$$

for 'm' authors if we do not use blocking, whereas, blocking considerably reduces computational complexity to

$$\mathcal{O}(L\,|T|) \tag{6}$$

Where 'L' is the number of blocks and 'T' is the average size of blocks.

If we set a very high threshold for blocking then it produces very low recall but pure blocks. In contrast, if we set very low threshold then it produces high false positives. So, we sampled names data set and set this threshold to 0.8 that produces optimal blocking of ambiguous author names.

Citations data set provides some basic features like names of authors, titles, venues and year of publications. However, as mentioned earlier in "Introduction", the most influential feature among these is the co-authors for solving the problem of author ambiguity [4, 18, 19, 21]. In LUCID, an author is represented by a node that has its unique id for identification, its name (not

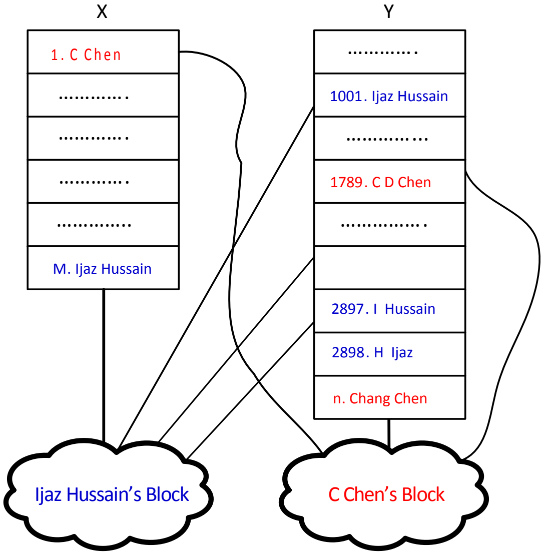


Fig. 2. An example of ambiguous author blocking.

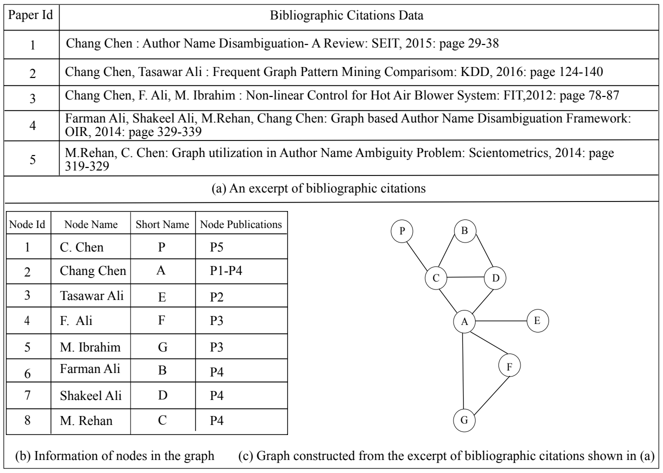
unique), and publications set where that author has appeared. The edges modeled the bidirectional relationship between two authors. Every citation that has 'm' number of authors(nodes) exactly produces 'E' number of edges, given by (7). A toy example of citations data set, nodes and constructed graph is shown in Fig. 3. There are eight author nodes and ten edges that are created from five citations. Chang Chen is a node that is common to four citations and in one citation it's abbreviated name "C Chen" is mentioned.

$$E = \frac{(m) * (m-1)}{2} \tag{7}$$

C. Homonyms Resolver Stage

It is assumed here that different homonyms have different social circles and different authors with the same name seldom work in the same organization or social circle [19, 18]. So, they should form quite different author communities. A community in the co-author's graph of LUCID is generated from each citation and thus each community denotes the co-authors for each citation that is the smallest social circle in the academic domain. With the help of co-authors graph, it is possible to infer a social circle of an author from his/her co-authors by finding communities. The community gets bigger and wider when more citations are processed. LUCID assumes that an author linked with multiple social circles contains mixed information for several authors that happened to have the same name. LUCID splits that node and its information into a number of different nodes that are present in non-overlapping communities. LUCID considers each non-overlapping community emanating from the same node as a different social circle of an author. LUCID community detection and split algorithm pseudo-code is described in Algorithm 1.

In co-authors graph, an edge represents a co-authors relationship between two authors (nodes). LUCID's community detection algorithm is based on "SCAN: A Structural Clustering Algorithm for Networks" to detect different communities in the co-author's graph [27]. When SCAN is applied on co-authors graph it outputs communities (clusters of nodes), hub node and outliers as defined in "Preliminaries". The community consists of a set of nodes from this co-authors graph. The node that is involved in many social circles in the co-author's graph is called hub node. Hub node is the potential homonym that needs disambiguation. Outliers are nodes that only contribute one or a few publications with hub node. In a toy graph example, detected communities, hub and outlier can be seen in Fig. 4. There are two clusters of nodes (communities), one hub and one outlier. The first and second cluster consists of nodes ⟨P, B, C, D⟩ and ⟨F, G⟩ respectively. The hub is node ⟨A⟩ and outlier is node ⟨E⟩.



Paper Id	Bibliographic Citations Data
1	Chang Chen : Author Name Disambiguation- A Review: SEIT, 2015: page 29-38
2	Chang Chen, Tasawar Ali : Frequent Graph Pattern Mining Comparisom: KDD, 2016: page 124-140
3	Chang Chen, F. Ali, M. Ibrahim : Non-linear Control for Hot Air Blower System: FIT,2012: page 78-87
4	Farman Ali, Shakeel Ali, M.Rehan, Chang Chen: Graph based Author Name Disambiguation Framework: OIR, 2014: page 329-339
5	M.Rehan, C. Chen: Graph utilization in Author Name Ambiguity Problem: Scientometrics, 2014: page 319-329

(a) An excerpt of bibliographic citations

Node Id	Node Name	Short Name	Node Publications
1	C. Chen	P	P5
2	Chang Chen	A	P1-P4
3	Tasawar Ali	E	P2
4	F. Ali	F	P3
5	M. Ibrahim	G	P3
6	Farman Ali	B	P4
7	Shakeel Ali	D	P4
8	M. Rehan	C	P4

(b) Information of nodes in the graph (c) Graph constructed from the excerpt of bibliographic citations shown in (a)

Fig. 3. Co-authors graph of Chang Chen sample citations.

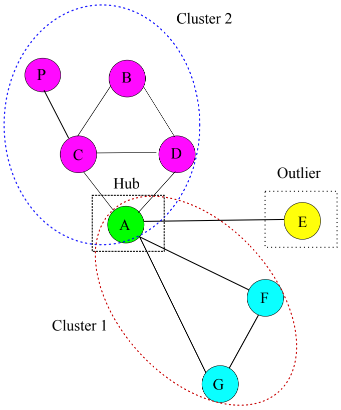


Fig. 4. An example of detected communities, hub and outlier after using SCAN on Chang Chen co-authors graph.

A hub node that has multiple non-overlapping communities contains mixed information for several homonym authors. LUCID splits the information of each author on the node containing the homonyms along the non-overlapping communities. This part starts from the line 2 of Algorithm 1, where the hub node publications list is retrieved from the graph. Similarly, publications of all cluster nodes (community) are retrieved and saved in a list (line 3-5). Now, for each homonym the intersection of hub publications and community publications

Algorithm 1: LUCID algorithm

```
Data: $Co - authors\ Graph(CG)$
Result: $Disambiguated\ Graph(DG)$
begin
1   $[Clusters, Hub, Outliers] \longleftarrow Apply\_SCAN(CG)$
2   $Hub\_Publications \longleftarrow get\_Publications(Hub)$
3   for $cluster \in Clusters$ do
4       for $node \in cluster$ do
5           $cluster\_Publications \longleftarrow cluster\_Publications \cup get\_Publications(node)$
6       $HN_i\ Publications \longleftarrow Hub\_Publications \cap cluster\_Publications$
7       $HN_i\ Name \longleftarrow get\_Name(Hub)$
8       $HN_i\ Id \longleftarrow get\_Length(CG)$
9       $Insert\_Node(HN_i\ Id, HN_i\ Name, HN_i\ Publications)$
10      $Update\_Graph(CG, cluster)$
11      $Hub\_Publications \longleftarrow Hub\_Publications \setminus cluster\_Publications$
12      $Update\_Graph(CG)$
13  $Hub\_Name \longleftarrow HG.getName(Hub)$
14  for $node \in HG$ do
15      $all\_Author\_Names \longleftarrow all\_Author\_Names \cup HG.getName(node)$
16  $comparison\_Name \longleftarrow Hub\_Name$
17  $name\_Tokens \longleftarrow comparison\_Name.split()$
18  $all\_Author\_Names.delete(Hub\_Name)$
19  for $author \in all\_Author\_Names$ do
20      $similarity \longleftarrow name\_Similarity(comparison\_Name, author)$
21      if $similarity > threshold\ \&\ author\_compatible$ then
22          $similar\_Names\_List.append(author)$
23  for $name \in similar\_Names\_List$ do
24      $geodesic\_Distance \longleftarrow geodesic\_Distance(comparison\_Name, name)$
25      if $geodesic\_Distance < threshold$ then
26          $mergeNodePubs \longleftarrow comparison\_Name\_Publications \cup name\_Publications$
27          if $name\_Id < comparison\_Name\_Id$ then
28              $mergeNodeName \longleftarrow name\_Name$
29              $mergeNodeId \longleftarrow name\_Id$
30          else
31              $mergeNodeName \longleftarrow comparison\_Name$
32              $mergeNodeId \longleftarrow comparison\_Id$
33          $Insert\_New\_Node \longleftarrow (mergeNodeId, mergeName, mergeNodePubs)$
34          $deleteNode(comparison\_Node)$
35          $Update\_Graph(HG)$
36      $return\_Updated\_Graph\ (DG)$
```

are found. A new node is created in the co-author's graph for each community detected that has the same name as that of the hub node and has identity one more than in the current nodes in the co-author's graph, this newly created node has the publication list that is found at line 6. Likewise, for all communities detected in the co-author's graph, new nodes are created and the graph is updated after every new node is inserted in the co-author's graph. Publications of hub node are also updated after every insertion. In graph update, some edges to the hub node to existing communities are removed and some new edges to newly created nodes are created (line 7-12). The same procedure is repeated for all outliers, as is done in the case of all communities. This whole process is pictorially shown in Fig. 5.

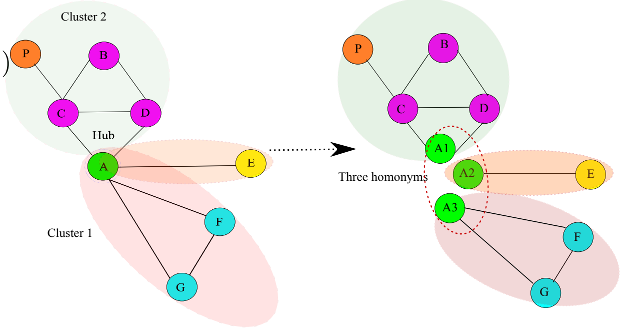


Fig. 5. An example of homonyms resolution.

As seen in it hub node 'A' is split into three new nodes ⟨ A1, A2, A3⟩, as there are two clusters of nodes (communities) and one outlier. In this way, the homonym problem is solved using community detection algorithm and graph operations.

D. Synonyms Resolver

The second type of name ambiguity is the synonym that occurs when an author has a similar name as explained in the subsequent paragraphs. In academic domain, authors often fill out their names in different forms–abbreviation, sequence changing, an omission of some part of their name, etc. So, the same author may appear in literature with many different forms of his name.

Co-authors graph indicates the author's past and present connections. So, if apparently different researchers having similar names and there is a small geodesic distance between them then it is probably the same author. In LUCID, the nodes that have similar names referred as the same person if they have a geodesic distance less than 3, meaning that they are having very close relationship with the similar author. First, LUCID searches the whole graph for nodes with similar author names. The names are considered to be similar if they have a syntactic similarity greater than a predefined threshold and compatible names. The Jaro-Winkler similarity is used for this purpose as it proved to be the best similarity measure for names [25]. Compatible names are those names that are either fully part of another name or they have the same initial of the first and last name. For example, a name "C Chen" is fully part of another name "C Chen Chang" and a name "Lei Wang" is not having the same initials as that of another name "Wei Wang". Similarly, another pair of names is "A. Gupta" and

"Anuram Gupta" that have same initial. So, the first and third pair of names are compatible and the second pair of names is not compatible. Pseudo-code for searching similar names is described in Algorithm 1 (lines 19-22).

After finding similar names the geodesic distance between two similar name nodes is found. Those author nodes whose geodesic distance is less than the threshold are considered the synonyms. Two synonym nodes are then merged into one node having publication list of both of these two nodes and id of the full name is preserved in the graph whereas the other similar name node is deleted from the graph. Pseudo-code for merging similar names is described in Algorithm 1 (lines 23-35).

An example of synonym merging is shown in Fig. 6. In this example, since node 'P' and node 'A1' have syntactic similarity less than a threshold and are compatible names. Furthermore, the geodesic distance between these nodes is less than the threshold. So in this case, they are merged into one node 'A1'and the other node 'P' is deleted from the co-authorship graph and the edges are updated.

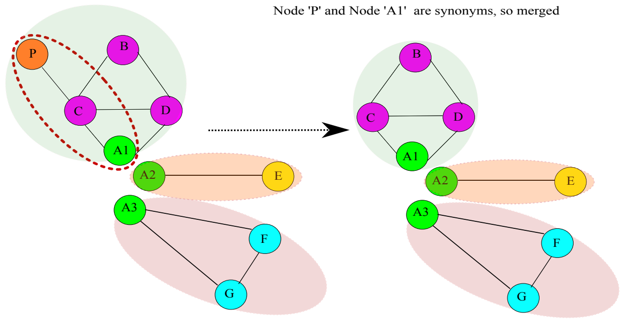


Fig. 6. An example of synonyms resolution.

IV. Experiments and Discussions

For the HHC and GFAD methods, we used the implementation of [2] and [18], respectively. All experiments were performed on the Intel (R) 64-bit Core (TM) i5-5200U 2x2.2 GHz processors with 8 GB of RAM.

A. Data set

A real world publications data set from Arnetminer is used to measure the effectiveness of LUCID. The statistics of the Arnetminer data set are given in Table I and more details are found online at Arnetminer. The original version of this collection was created by Wang et al. [24]. Then Tang et al. [19] manually checked, labeled and included more ambiguous authors to expand this data set. It is used with slight variations in many AND studies [19, 18, 26]. Subsets of this data set have also been used in many earlier studies [8, 9, 2, 6, 5].

In LUCID, only co-authors information of this data set is used for the complete solution of the author name ambiguity problem. LUCID considered two authors same if their full name are identical.

B. Evaluation Metrics

Pairwise Precision (PP), Recall (PR) and F1 (PF1) are used to measure the effectiveness of LUCID. PF1 is the harmonic mean of PP and PR. The fraction of citation records corresponding to the same author in the system-generated clusters are known as PP and fraction of citation records associated with the same author in the gold standard reference clusters are called as PR. The PP, PR and PF1 measures are expressed in (8), where $C(n,i)$ denotes the number of combinations of i elements from n elements: $C(n,i) = \frac{n!}{i!(n-i)!}, n \geq i$. Here, n denotes the size of the citations in the collection, j is the number of gold standard reference clusters manually generated, and i is the number of clusters automatically generated by the LUCID. Also, n_j is the number of elements in cluster j and n_{ij} is the number of elements belonging to both clusters i and j.

$$PP = \frac{\sum_{i=1}^{I}\sum_{j=1}^{J} C(n_{ij},2)}{\sum_{i=1}^{I} C(n_i,2)}, PR = \frac{\sum_{i=1}^{I}\sum_{j=1}^{J} C(n_{ij},2)}{\sum_{j=1}^{J} C(n_j,2)}, PF1 = \frac{2 * PP * PR}{PP + PR} \quad (8)$$

TABLE I. The Arnetminer Data Set (Auth. Denotes Number of Distinct Authors and Rec. Represents Citations Records Associated with that Author)

Name	Auth.	Rec.	Name	Auth.	Rec.
Bin Li	8	31	B. Wilkinson	1	17
Bin Zhu	15	45	Cheng Chang	5	23
Charles Smith	4	7	Paul Wang	6	15
Michael Siege	6	50	David Cooper	7	18
Gang Luo	8	42	S Huang	15	16
J. Guo	9	12	Hui Yu	20	30
Yan Tang	11	31	Lei Fang	7	17
Xiaoming Wang	14	33	Yue Zhao	9	36
Paul Brown	8	20	Peter Phillips	3	13
David Nelson	10	15	F. Wang	14	15
Hong Xie	6	10	J. Yin	7	18
John Hale	3	36	Kuo Zhang	4	16
Michael Smith	16	27	Richard Taylor	11	27

TABLE II. The Performance Evaluation of LUCID for Each Ambiguous Group on the Arnetminer Collection

Name	PP	PR	PF1	Name	PP	PR	PF1
Bin Li	1.00	0.64	0.80	Hui Yu	1.00	0.97	0.98
Barry Wilkinson	1.00	1.00	1.00	J. Yin	0.95	0.52	0.70
J. Guo	1.00	0.85	0.92	S. Huang	1.00	0.63	0.80
Bin Zhu	1.00	1.00	1.00	John Hale	1.00	0.52	0.72
Charles Smith	1.00	1.00	1.00	Kuo Zhang	1.00	0.91	0.95
Cheng Chang	1.00	0.79	0.89	Lei Fang	1.00	0.90	0.95
David Cooper	1.00	0.92	0.96	Yue Zhao	1.00	1.00	1.00
David Nelson	1.00	0.93	0.97	Peter Phillips	1.00	0.79	0.89
F. Wang	1.00	1.00	1.00	Richard Taylor	1.00	0.73	0.86
Yan Tang	0.95	0.71	0.83	Michael Siege	1.00	0.83	0.91
Gang Luo	1.00	0.95	0.98	Michael Smith	1.00	0.76	0.87
Hong Xie	1.00	0.87	0.93	Paul Brown	1.00	1.00	1.00
Xiaoming Wang	1.00	0.70	0.83	Paul Wang	1.00	0.91	0.95
				Average	0.99	0.84	0.91

C. Baseline Methods

Two unsupervised author name disambiguation methods–heuristic based hierarchical clustering (HHC) [2] and author name disambiguation using a graph model (GFAD) [18] were used as baseline methods to compare with LUCID.

HHC is a heuristic based unsupervised method that uses citation features for author name disambiguation. HHC showed the best results as compared with other unsupervised and supervised methods [3]. We compared the performance of LUCID with the HHC using all the features of citations (co-authors, titles, and venue) that they called HHC-All in their paper. HHC-All needs similarity thresholds for the paper title and

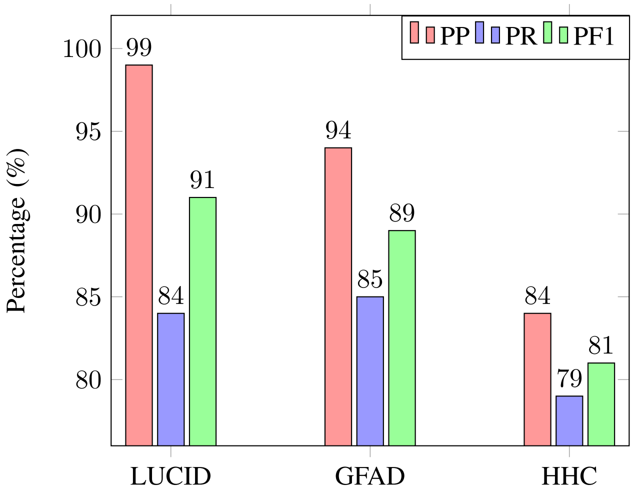


Fig. 7. Average pairwise precision, pairwise recall and pairwise F1 of the three techniques.

publication venue. Therefore, we evaluated the performance of HHC-All by varying threshold values and then selected values on which the performances are maximized. Optimal results are achieved at 0.2 and 0.3 similarity threshold values for paper title and publication venue respectively. For further details on HHC, refer to [2]. To compare LUCID with GFAD, we use the disambiguation results of the GFAD as given in [18] as it is because LUCID also uses the same data set and same performance metrics as GFAD.

D. Performance Evaluation

LUCID performance is evaluated using three well-known clustering metrics on Arnetminer collection of citation records described in Section IV-B. It achieves on average PP of 99%, PR of 84% and PF of 91% on Arnetminer data set. Table II lists the complete details of each ambiguous author group in the Arnetminer collection. In general PP of LUCID in the majority of cases are approximately equal to 100% but in some cases such as “Yan Tang” and “J. Yin” is low. In Fig. 7, LUCID performance is compared with two baseline methods–HHC and GFAD. It shows overall better performance than baseline methods. It achieved 5 and 2% higher in PP and PF1 than GFAD. Similarly, it achieved 15 and 11% higher in PP and PF1 than HHC. GFAD performed better in PR by 1% higher than LUCID.

LUCID performance is overall better as compared to baseline methods due to the reason that this takes the advantage of semantic relationships between authors. GFAD is unable to detect outlier authors that only share one co-author whereas LUCID is able to identify these authors. HHC only takes into account the syntax similarities of attributes not semantics.

Comparison between ground truth clusters and LUCID generated clusters is shown in Fig. 8. About 70% clusters are within the range (Ground Truth Clusters±4). However, in the majority of cases, LUCID produced more clusters than the ground truth clusters. This is due to the fact that many authors have multiple affiliations and research interests in their entire career and thus have different social circles.

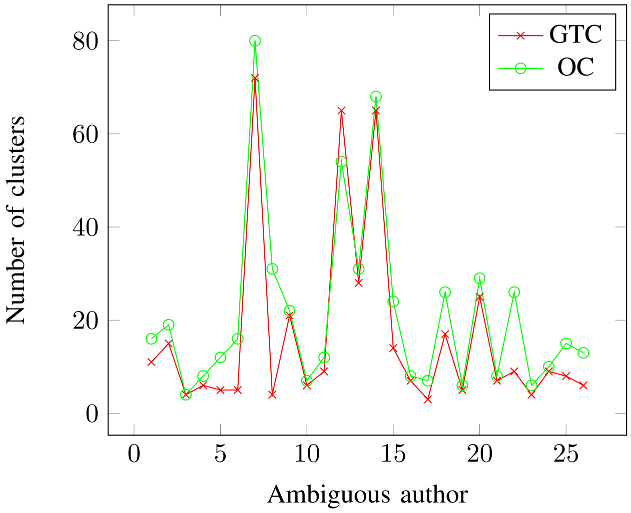


Fig. 8. Comparison of ground truth clusters and LUCID generated clusters (GTC denotes ground truth clusters and OC refers to LUCID generated clusters).

LUCID performance is better than baseline methods in all metrics except in PR where it is 1% lower than GFAD. However, LUCID still has some limitations: 1) it is unable to identify when two different authors with the same name have different co-authors with the same name; 2) it also fails to disambiguate when two citation records do not share common co-authors although they might be written by the same authors; 3) it may not resolve authors correctly when an author changes his/her area of research and works with entirely different set of co-authors.

V. Conclusion

Community detection algorithms and hybrid similarity are little used in the domain of author name disambiguation. In this paper, we have proposed LUCID that solves author name ambiguity problem using only co-authors. LUCID is a graph-based method that does not require costly training data or a priori or hidden information (how many ambiguous authors) and web searches, solves both homonym and synonym problem. In this study, we present homonym resolver algorithm and synonym resolver algorithm as our main contributions.

LUCID performance was tested on Arnetminer, a real world data set and it showed better results than two baseline studies HHC and GFAD. LUCID delivered a simple but effective solution to author name ambiguity problem as it utilized the powers of graph-based algorithms. However, it is unable to detect very ambiguous author cases where two different authors with the same name work with two different co-authors with the same name and in the cases where a researcher has multiple different research interests and is working separately on these research interests with completely different set of co-authors. In the future, we plan to analyze self-citations, hidden concepts and email address of authors to overcome these limitations.

References

[1] Indrajit Bhattacharya and Lise Getoor. Collective entity resolution in relational data. *ACM Transactions on Knowledge Discovery from Data*

(TKDD), 1(1):5, 2007.

[2] Ricardo G Cota, Anderson A Ferreira, Cristiano Nascimento, Marcos André Gonçalves, and Alberto HF Laender. An unsupervised heuristic-based hierarchical method for name disambiguation in bibliographic citations. *Journal of the American Society for Information Science and Technology*, 61(9):1853–1870, 2010.

[3] Ana Paula De Carvalho, Anderson A Ferreira, Alberto HF Laender, and Marcos A Gonçalves. Incremental unsupervised name disambiguation in cleaned digital libraries. *Journal of Information and Data Management*, 2(3):289, 2011.

[4] Xiaoming Fan, Jianyong Wang, Xu Pu, Lizhu Zhou, and Bing Lv. On graph-based name disambiguation. *Journal of Data and Information Quality (JDIQ)*, 2(2):10, 2011.

[5] Anderson A Ferreira, Marcos André Gonçalves, and Alberto HF Laender. A brief survey of automatic methods for author name disambiguation. volume 41, pages 15–26. ACM, 2012.

[6] Anderson A Ferreira, Adriano Veloso, Marcos André Gonçalves, and Alberto HF Laender. Effective self-training author name disambiguation in scholarly digital libraries. In *Proceedings of the 10th annual joint conference on Digital libraries*, pages 39–48. ACM, 2010.

[7] Donghong Han, Siqi Liu, Yachao Hu, Bin Wang, and Yongjiao Sun. Elm-based name disambiguation in bibliography. *World Wide Web*, 18(2):253–263, 2015.

[8] Hui Han, Lee Giles, Hongyuan Zha, Cheng Li, and Kostas Tsioutsiouliklis. Two supervised learning approaches for name disambiguation in author citations. In *Digital Libraries, 2004. Proceedings of the 2004 joint ACM/IEEE conference on*, pages 296–305. IEEE, 2004.

[9] Jianbin Huang, Heli Sun, Qinbao Song, Hongbo Deng, and Jiawei Han. Revealing density-based clustering structure from the core-connected tree of a network. *IEEE Transactions on Knowledge and Data Engineering*, 25(8):1876–1889, 2013.

[10] Tin Huynh, Kiem Hoang, Tien Do, and Duc Huynh. Vietnamese author name disambiguation for integrating publications from heterogeneous sources. In *Asian Conference on Intelligent Information and Database Systems*, pages 226–235. Springer, 2013.

[11] Muhammad Imran, Syed Gillani, and Maurizio Marchese. A real-time heuristic-based unsupervised method for name disambiguation in digital libraries. *D-Lib Magazine*, 19(9):1, 2013.

[12] Felipe Hoppe Levin and Carlos A Heuser. Evaluating the use of social networks in author name disambiguation in digital libraries. *Journal of Information and Data Management*, 1(2):183, 2010.

[13] Yuechang Liu and Yong Tang. Network based framework for author name disambiguation applications. *International Journal of u-and e-Service, Science and Technology*, 8(9):75–82, 2015.

[14] Eamonn James Maguire. Ethnicity sensitive author disambiguation using semi-supervised learning. In *Knowledge Engineering and Semantic Web: 7th International Conference, KESW 2016, Prague, Czech Republic, September 21-23, 2016, Proceedings*, volume 649, page 272. Springer, 2016.

[15] Byung-Won On, Ingyu Lee, and Dongwon Lee. Scalable clustering methods for the name disambiguation problem. *Knowledge and Information Systems*, 31(1):129–151, 2012.

[16] Natsuo Onodera, Mariko Iwasawa, Nobuyuki Midorikawa, Fuyuki Yoshikane, Kou Amano, Yutaka Ootani, Tadashi Kodama, Yasuhiko Kiyama, Hiroyuki Tsunoda, and Shizuka Yamazaki. A method for eliminating articles by homonymous authors from the large number of articles retrieved by author search. *Journal of the American Society for Information Science and Technology*, 62(4):677–690, 2011.

[17] Hsin-Tsung Peng, Cheng-Yu Lu, William Hsu, and Jan-Ming Ho. Disambiguating authors in citations on the web and authorship correlations. *Expert Systems with Applications*, 39(12):10521–10532, 2012.

[18] Dongwook Shin, Taehwan Kim, Joongmin Choi, and Jungsun Kim. Author name disambiguation using a graph model with node splitting and merging based on bibliographic information. *Scientometrics*, 100(1):15–50, 2014.

[19] Jie Tang, Alvis CM Fong, Bo Wang, and Jing Zhang. A unified probabilistic framework for name disambiguation in digital library. *IEEE Transactions on Knowledge and Data Engineering*, 24(6):975–987, 2012.

[20] Li Tang and John P Walsh. Bibliometric fingerprints: name disambiguation based on approximate structure equivalence of cognitive maps. *Scientometrics*, 84(3):763–784, 2010.

[21] Hung Nghiep Tran, Tin Huynh, and Tien Do. Author name disambiguation by using deep neural network. pages 123–132, 2014.

[22] Pucktada Treeratpituk and C Lee Giles. Disambiguating authors in academic publications using random forests. In *Proceedings of the 9th ACM/IEEE-CS joint conference on Digital libraries*, pages 39–48. ACM, 2009.

[23] Jian Wang, Kaspars Berzins, Diana Hicks, Julia Melkers, Fang Xiao, and Diogo Pinheiro. A boosted-trees method for name disambiguation. *Scientometrics*, 93(2):391–411, 2012.

[24] Xuezhi Wang, Jie Tang, Hong Cheng, and S Yu Philip. Adana: Active name disambiguation. In *2011 IEEE 11th International Conference on Data Mining*, pages 794–803. IEEE, 2011.

[25] William E Winkler. String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage. 1990.

[26] Hao Wu, Bo Li, Yijian Pei, and Jun He. Unsupervised author disambiguation using dempster-shafer theory. *Scientometrics*, 101(3):1955–1972, 2014.

[27] Xiaowei Xu, Nurcan Yuruk, Zhidan Feng, and Thomas AJ Schweiger. Scan: a structural clustering algorithm for networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 824–833. ACM, 2007.

[28] Jia Zhu, Yi Yang, Qing Xie, Liwei Wang, and Saeed-Ul Hassan. Robust hybrid name disambiguation framework for large databases. *Scientometrics*, 98(3):2255–2274, 2014.