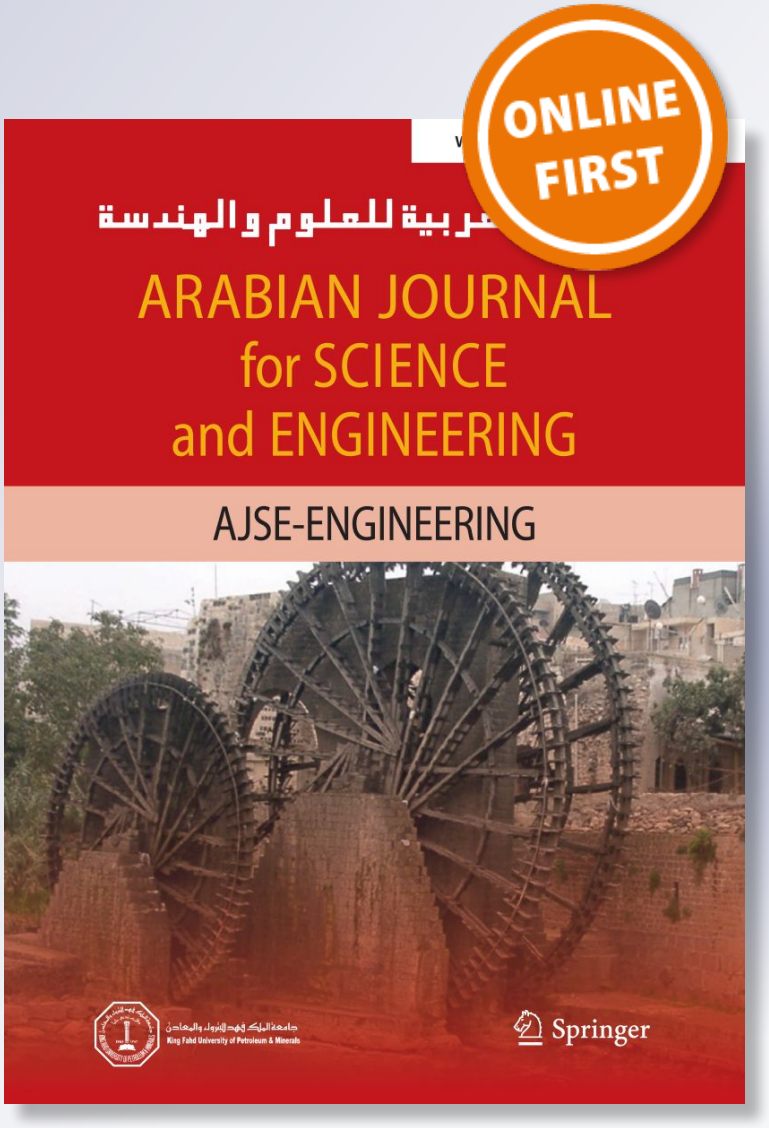
Author Name Disambiguation by Exploiting Graph Structural Clustering and Hybrid Similarity

Ijaz Hussain & Sohail Asghar

Arabian Journal for Science and Engineering

ISSN 2193-567X

Arab J Sci Eng
DOI 10.1007/s13369-018-3099-0



Springer

Your article is protected by copyright and all rights are held exclusively by King Fahd University of Petroleum & Minerals. This e-offprint is for personal use only and shall not be self-archived in electronic repositories. If you wish to self-archive your article, please use the accepted manuscript version for posting on your own website. You may further deposit the accepted manuscript version in any repository, provided it is only made publicly available 12 months after official publication or later and provided acknowledgement is given to the original source of publication and a link is inserted to the published article on Springer's website. The link must be accompanied by the following text: "The final publication is available at link.springer.com".

Author's personal copy

Arabian Journal for Science and Engineering
https://doi.org/10.1007/s13369-018-3099-0

RESEARCH ARTICLE - COMPUTER ENGINEERING AND COMPUTER SCIENCE

Author Name Disambiguation by Exploiting Graph Structural Clustering and Hybrid Similarity

Ijaz Hussain[1] · **Sohail Asghar**[1]

Received: 16 March 2017 / Accepted: 4 February 2018
© King Fahd University of Petroleum & Minerals 2018

Abstract
Author name ambiguity occurs when multiple authors share a common name and an author writes one's name in many ways. This hinders the quality of information retrieval and correct attribution to authors in bibliographic databases. Despite much research in the past decade, the author name ambiguity problem remains largely unsolved. Outstanding issues include limited capabilities (solve only homonyms or synonyms), require extra information (Web or user feedback), actual number of authors K in advance and not scalable. In this paper, a method called GCLUSIM is proposed which uses graph structural clustering and proposed similarity measure to resolve ambiguous authors. GCLUSIM preprocesses citation data set and constructs co-authors graph. Graph-based structural clustering is applied to the constructed graph to identify hub nodes, outliers, and clusters of nodes. It resolves homonyms by splitting these clusters if the feature vector similarity between these clusters is less than the predefined threshold and synonyms by exploiting proposed similarity. Finally, it disambiguates sole authors by comparing name and feature vector similarities with the disambiguated clusters. Experiments are performed with Arnetminer and BDBComp to validate the performance of the GCLUSIM. Results show that GCLUSIM is scalable, overall better in performance than baselines and the number of clusters found is close to the ground truth clusters.

Keywords Author name disambiguation · Co-authors graph · Homonym resolver · Synonyms resolver · Sole author resolver

1 Introduction

Several people in the world may have exactly the same name, and it is also possible that one may write one's name in many different ways—abbreviations, changing the order and omitting some name part. The former is called homonym, and the latter is called synonym. For example, when we search in the DBLP (Digital Bibliography & Library Project) an author name "Wei Wang", it turns out eighty-five different authors that have exactly the same name "Wei Wang". Similarly, when we search an author name "W. Wang", it turns out more than a thousand likely matches. More or less this is the situation in all bibliographic databases. This is considered a subproblem of entity resolution problem among information retrieval research community [1–3]. Author name ambiguities can cause wrong attributions and incorrect search results [4–6]. The methods that resolve these author name ambiguities are called Author Name Disambiguation (AND) systems.

Numerous AND methods were proposed in the last decade that used machine learning and data mining techniques [1–15]. However, many of these dealt only with the homonyms [6,12,16], some of them disambiguated only the synonyms [17] and few others heavily relied on ancillary evidence such as email, affiliation or related web pages, or required some hidden information such as the number of ambiguous authors or groups [2,13,18]. Some others required costly training data to train their model parameters [9,19,20]. Still others were not scalable and produced false positives that severely affect their performance [5,12,16]

In this paper, we propose "Author Name Disambiguation by Exploiting Graph Structural Clustering and Hybrid Similarity Index" called GCLUSIM, to resolve these problems. GCLUSIM is a graph-based technique in that citation data set is preprocessed and ambiguous author blocks are identified, as given in [8]. Co-authors graph is constructed using the preprocessed citation data and "SCAN: A Structural Clustering Algorithm for networks" is used to identify hub nodes, outliers, and clusters of nodes [21]. SCAN is used due to three

✉ Ijaz Hussain
fa14-pcs-006@student.comsats.edu.pk

[1] Department of Computer Science, COMSATS Institute of Information Technology, Islamabad, Pakistan

Published online: 16 February 2018

Author's personal copy

reasons, in contrast to other graph methods, it is able to detect outliers, is scalable as its complexity is $\mathcal{O}(e)$, where e is the number of edges in co-authors graph and is capable of identifying overlapping communities unlike other community detection algorithms. Homonyms are resolved by splitting identified clusters of nodes if the similarity between these clusters is less than a predefined threshold, details are given in Sect. 3.2. Synonyms are solved using proposed hybrid similarity index. When two names are compatible and their hybrid similarity index is greater than a predefined threshold, then these nodes are merged into one node, as described in Sect. 3.3. Finally, sole authors are disambiguated by comparing the similarity between feature vectors of two similar names. GCLUSIM only uses the co-authors information and abstracts to fully resolve the author name ambiguity problems. GCLUSIM simultaneously resolves homonyms and synonyms. In the last, step sole authors are resolved that many previous studies did not address [6,12,17].

GHOST presented by [12] used valid path-based similarity between two nodes that is in the order of

$$\mathcal{O}((|R^2|) \times (|V| + |E|)) \tag{1}$$

where V is the vertices, E is the edges and R is the set of names to resolve. Another method ADANA given by [16] used pairwise factor graph that was also intractable due to the calculation of normalization factor that made their method complexity to exponential to the number of nodes in the graph. Outliers are nodes that have only one or a few citations common with ambiguous author; details are presented in Sect. 3.2. Similarly GFAD proposed by [5] used Johnson's simple cycle enumeration algorithm [22] that has complexity in the order of

$$\mathcal{O}((n + e)(c + 1)) \tag{2}$$

for n vertices, e edges and c elementary cycles in the graph. In the unified probabilistic framework, [3] transformed the content and structure information about citation attributes into the Hidden Markov Random Fields as feature functions and formed an objective function to find the hidden parameters. However, this method worked well on small ambiguous author groups. As the number of ambiguous authors increased, it was unable to correctly predict the number of ambiguous authors *'k'*.

To evaluate the performance of GCLUSIM, we compare it with three unsupervised name disambiguation methods [5, 12,23] using collection of citations from two real-world bibliographic databases Arnetminer and BDBComp. GCLUSIM performance is overall better than these systems from the perspective of representative clustering evaluation metrics despite the fact that GCLUSIM uses only the co-authors and abstract. In summary, the main contributions of this work are:

- Graph structural clustering is employed to solve the homonyms in contrast to the existing techniques that either used cycle enumerations or graph paths. To the best of our knowledge, the identification of outliers for the author name ambiguity in graph-based methods has been addressed for the first time,
- A hybrid similarity index is proposed, to resolve the synonyms, in which information of syntactic similarity and graph geodesics between vertices is exploited,
- A complete author name disambiguation method, "GCLUSIM," is presented that requires no costly training data, no hidden information about number of unknown clusters, resolves sole authors and simultaneously solves both author name ambiguity problems, and
- Extensive experiments on real-world data sets of Arnetminer and BDBComp are performed to demonstrate the effectiveness of the GCLUSIM in Sect. 4.4.

The rest of this paper is structured as follows. Some related works are outlined, critically reviewed and analyzed at Sect. 2. In Sect. 3, GCLUSIM architecture and its working is discussed in detail. We have tested this approach on real-world data sets from Arnetminer and BDBComp to validate its performance against baselines, as given at Sect. 4. We conclude with a discussion of contributions and future research directions in Sect. 5.

2 Related Work

Existing AND methods can be categorized as supervised [6,20,24], unsupervised [3,10,13,25], semi-supervised [26], and graph-oriented approaches using graph models or social networks [5,12,16,27].

In [20], Wang et al. presented a boosted tree classification method in that they filter authors on the basis of name and affiliation matching. Similarity scores for different publication features were calculated, the false rate was used for author screening, and in the last stage boosted tree classifier was applied to the manually crafted data set. Two extreme learning machine-based algorithms for author name disambiguation were proposed in [6]. However, this method needed to train a new classifier model for every ambiguous name. A deep neural network-based approach to automatically learn the features and disambiguate the authors from the data set is proposed in [24] . They utilized the multicolumn deep neural network technique to improve the generalization capabilities of the system that is very similar to ensemble method bagging.

Cota et al. in [10] presented a two-step heuristic-based hierarchical clustering algorithm. They solved both homonyms and synonyms by exploiting similarities in citation attributes. The clusters were generated that had some

Author's personal copy

common co-author names among citations. This step generated fragmented but very pure clusters. Then these clusters were pairwise compared and merged if they have a similarity greater than some threshold in titles and venues. This process was iterative, and successive iterations had a synergistic effect on the disambiguation of authors, as more titles and venues were combined in merged clusters. A discrimination function that predicted true authors and false authors using logistic regression on Web of Science data was proposed in [13]. They extracted true author citations from a large amount of retrieved citations by using two stage filtering. This method is not good for citations whose subject fields and affiliation addresses do not vary too much. Wu et al. in [25] proposed an algorithm that used Dempster–Shafer Theory (DST) in combination with Shannon Entropy (SE) for author disambiguation. Some high-level features and their co-relation similarities were calculated and 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 the distance between clusters.

A unified probabilistic framework to address the homonyms was proposed in [3]. They formalized the disambiguation problem using Markov random fields. They automatically estimate the number of unknown ambiguous authors and the required unknown parameters. Zhu et al. [26] proposed a hybrid name disambiguation framework that not only used the implicit information, but also web page genre information. This framework consists of two main steps: web page genre identification and re-clustering model. A method called "GHOST" was presented in [12], in which they modeled the relationships among publications using undirected graphs. They solved homonym problem by iteratively finding valid paths, computing similarities, clustering with the help of affinity propagation algorithm and in the last step 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. Two multilevel graph partitioning algorithms for author name disambiguation were proposed in [23]. First is the multilevel graph partitioning algorithm (MGP) and the second the multilevel graph partitioning and merging (MGPM) algorithm. MGPM repeatedly merged the sub-graphs until the number of sub-graphs is equal to the given number of clusters (k) in the gold standard. This method requires the hidden information about the number of unknown authors in a cluster. Shin et al. in [5] proposed a graph framework for author name disambiguation "GFAD" that exploited co-authors and titles. 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. It resolves sole authors by comparing similarity among outliers with the help of cosine similarity measure. Shoaib et al. proposed a syntax-based similarity measure for AND in [28]. Some authors proposed ontologies to better search and retrieve data [29–31]. Recently, community detection algorithms are being used for information recommendation and in similar domains [32,33]. Semantic similarity measures and graph structural clustering are used in many domains such as Biomedical Informatics, GeoInformatics, Computational Linguistics, Community Detection, and in Natural Language Processing. Similarly, we are using graph-based semantic similarity measure and graph structural clustering in author name disambiguation domain because we believe it would prove to be useful in this domain.

GCLUSIM is different to other methods because it requires no training data while [6,20,24] required a lot of training data, it solves homonyms and synonyms simultaneously, whereas [3,12,13] solves only homonyms, it does not require user interaction as required in these methods [26,34], it does not need number of clusters K, and it solves sole authors, whereas [10,12,26] does not. GCLUSIM is congruent to methods that used community detection and semantic similarity in some different domains [29–33].

3 GCLUSIM Details

GCLUSIM architecture consists of following main stages—preparation and graph construction, homonym resolver, synonym resolver and sole authors resolver algorithms, as shown in Fig. 1. Details of all these stages are given in subsequent paragraphs.

3.1 Preparation and Graph Construction Stage

In the preprocessing, all the citation data set is extracted and tokened into authors, title, and venue terms. Stop words are removed from the titles and stemmed using Porter stemmer [35]. In blocking stage, similar names are grouped together in the candidate classes if their similarity is higher than some predefined threshold value. The blocking stage is important as it affects the computations in the later stages of the name disambiguation algorithms. Suppose if there are two author names lists of A and B, for every name $a \in A$, a set of author names $b_1, b_2, \ldots b_n \in B$ is found. In this way, the blocking stage returns M number of blocks if given N authors, as pictorially shown in Fig. 2. A block in author name disambiguation is all name variants which are potential candidates for ambiguous author names.

The computational complexity is $\mathcal{O}(N^2)$ for N authors if we do not use blocking, whereas, blocking considerably reduces computational complexity to $\mathcal{O}(R\,|S|)$, where R is

Author's personal copy

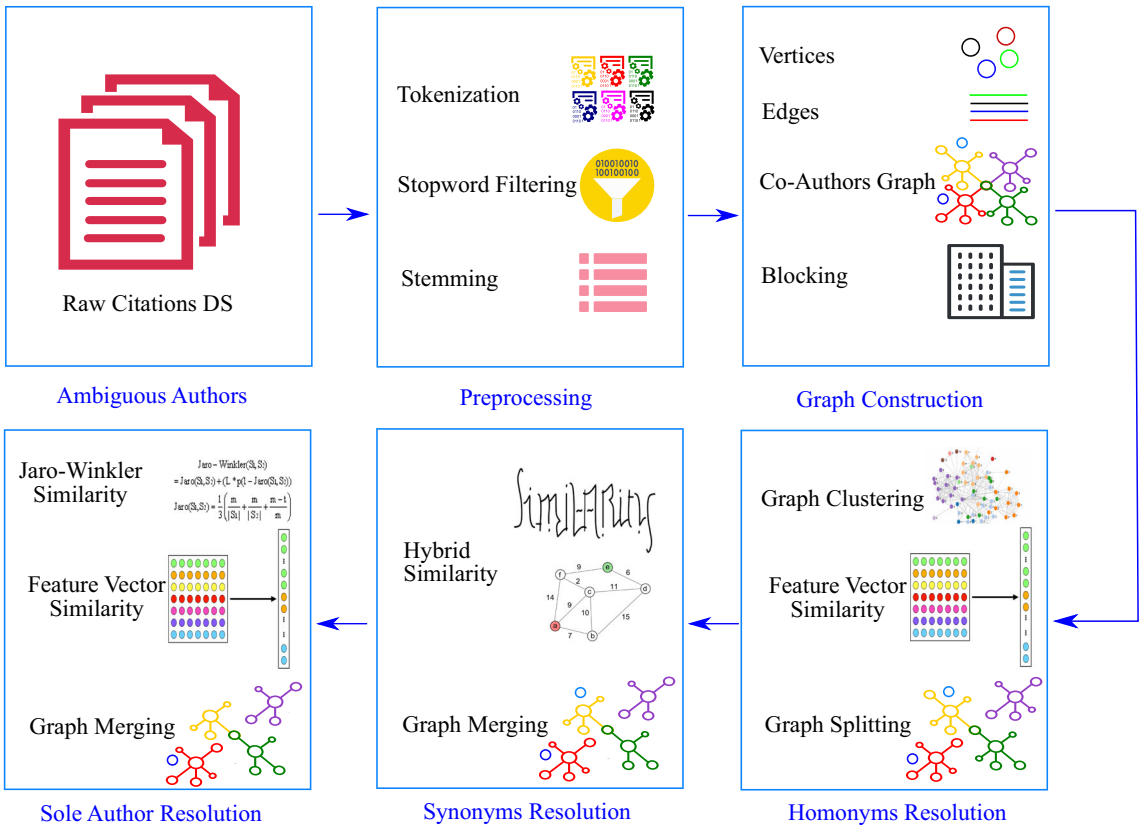


Fig. 1 The GCLUSIM Architecture

the number of blocks and S is the their size. If we set a very high threshold for blocking, then it produces low recall but very pure blocks. In contrast, if we set a very low threshold, then it produces high false positives. We randomly selected 10% similar name pairs, and 10% name variant pairs from each block of our ambiguous author data set and empirically found the threshold 0.8 that produces optimal author name blocks.

The citation data set provides basic features like names of authors, titles, venues and year of publications. However, as mentioned in the previous section, the most influential feature among these is the co-authors for solving the problem of author ambiguity [5,10,36]. In GCLUSIM, an author is represented by a node that has its unique id for identification, its name (not unique), and publications which that author has published. The edges modeled the bidirectional relationship between two co-authors. Every citation that has n number of authors (nodes) exactly produces E number of edges, as given in Eq. 3. An example citation data set is shown in Table 1.

Extracted nodes and constructed graph can be shown in Fig. 3, where there are twelve author nodes and twenty edges that are created from six citations. Chang Chen is a node that is common to five citations and in one citation it is a sole author, so an isolated node is created for it in the co-author's graph.

$$E = \frac{(n) * (n-1)}{2} \tag{3}$$

3.2 Proposed Homonym Resolver Algorithm

It is assumed that different homonyms have a different set of co-authors (social circles) and work in the related domain. Different authors with the same name seldom work in the same organization or social circle. So, they should form quite different author communities. In GCLUSIM, a community 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. For example, in 'citation 4' there are four co-authors B, D, C and A then the upper four nodes and their relationships are constructed as shown in Fig. 3 and when we add more citations to this graph, it becomes wider and bigger. Finally, when we construct the graph of all sample citations of Table 1, it looks like as depicted in Fig. 3. With the help of co-authors graph, it is possible to infer a social circle of an author from one's co-authors by finding shared communities.

Author's personal copy

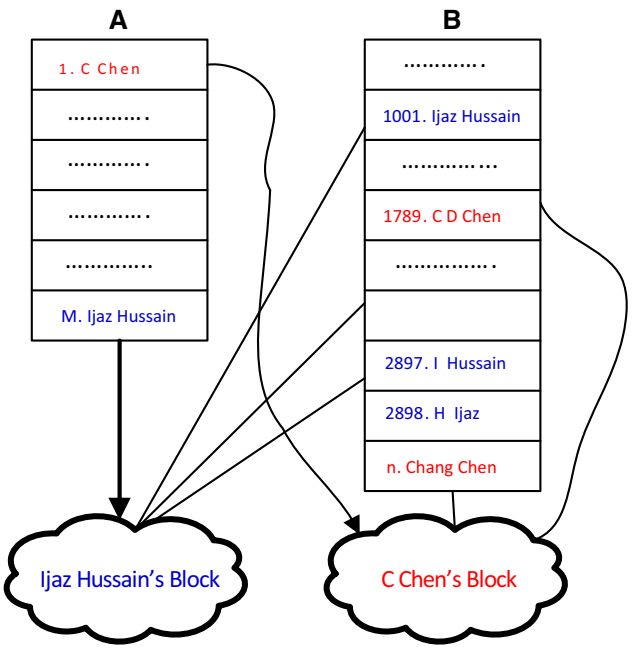


Fig. 2 Example of ambiguous author blocking

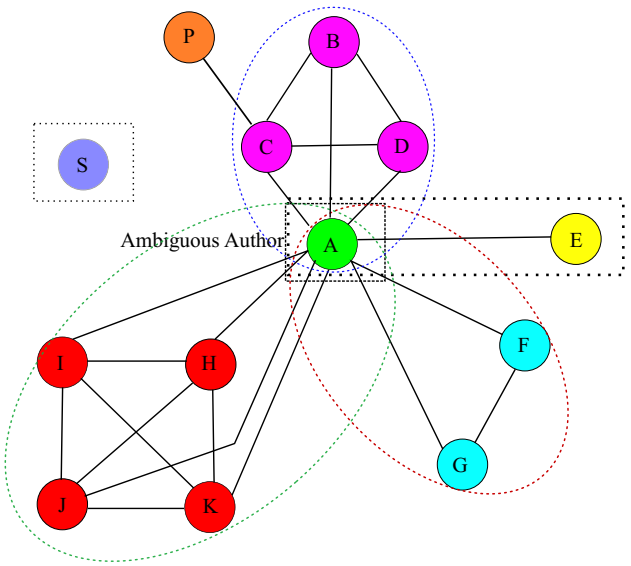


Fig. 3 An example of co-authors graph constructed from sample citation data set

In GCLUSIM we construct a feature vector from abstract keywords after removing stop words and then stemming these words. GCLUSIM assumes that an author linked with multiple social circles contains mixed information for several authors if the similarity between feature vectors of two clusters is < 0.2. This threshold is chosen empirically by varying its value in small increments and finding the optimal value of it. It is worthwhile to mention here that this step is needed only one time. GCLUSIM splits that node and its information into a number of different nodes that are present in non-overlapping communities. GCLUSIM considers each non-overlapping community emanating from the same node as a different social circle of an author if its feature vector similarity is less than the threshold. GCLUSIM community detection and split algorithm, pseudocode is described in Algorithm 1 and its details are given in Sect. 3.2.

3.2.1 Community Detector

In the co-authors graph, an edge represents a co-author relationship between the two authors. The GCLUSIM's community detection algorithm is based on "SCAN: Structural Clustering Algorithm for Networks" to detect different communities in the co-authors graph. This step is given in Algorithm 1 at line 1. In this section, we give brief description of necessary terms used in our method so that GCLUSIM can be understood in a self-contained fashion. However, interested readers are requested to read [21] for further technical details.

In homonym resolution algorithm, when we use SCAN on co-authors graph it outputs clusters of nodes (communities), hub nodes and outliers. The community consists of a set of nodes from this co-authors graph. We define hub as the node that is involved in many social circles in the co-authors graph. Hub node is the potential homonym that needs disambiguation. Outliers are nodes that only contribute one or

Table 1 An example citations data set

No.	Citations data	Author identities
1	Chang Chen : author name disambiguation—a review: SEIT, 2015: pages 29–38	S
2	Chang Chen, Tasawar Ali : frequent graph pattern mining comparison: KDD, 2016: pages 124–140	A , E
3	Chang Chen, F. Ali, M. Ibrahim : graph utilization in author name ambiguity problem: FIT, 2012: pages 78–87	A, F, G
4	Farman Ali, Shakeel Ali, M. Rehan, Chang Chen: graph-based author name disambiguation framework: OIR, 2014: pages 329–339	B , D, C, A
5	Chang Chen, Muaz Shah, Bilal Ahmad, Sanab Gul, Maham Ali : HAPT:HeArt failure prediction tool: FIT, 2016: page 295–308	A , I , J , K , H
6	M. Rehan, C. Chen: nonlinear control for hot air blower system: scientometrics, 2014: pages 319–329	C, P

Author's personal copy

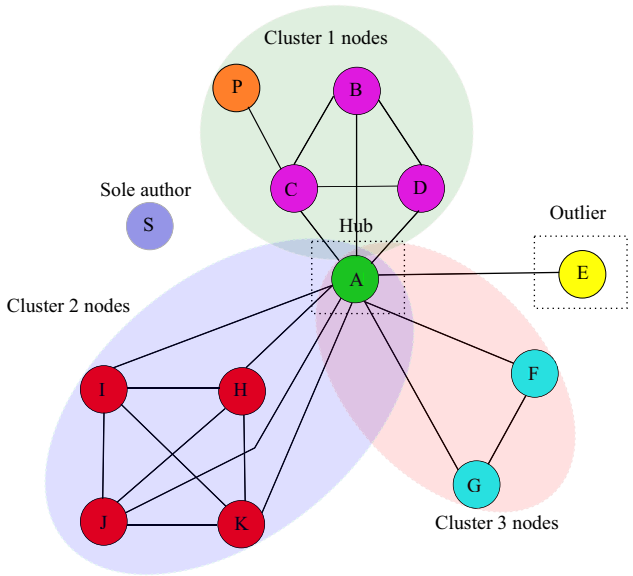


Fig. 4 An example of detected communities, hubs, and outliers after applying SCAN on co-authors graph

few publications with hub node. Clusters are vertices that are more similar to one another and different to other clusters. In example graph, detected communities, hub, and outlier can be seen in Fig. 4.

In this figure, there are three clusters of nodes (communities), one hub and one outlier. The first, second and third cluster consists of nodes (P, B, C, D), (H, I, J, K) and (F, G), respectively. The hub is node (A), and the outlier is node (E).

3.2.2 Community Splitter

A hub node that has multiple non-overlapping communities contains mixed information for several homonym authors. Feature vector of clusters is constructed as given in Algorithm 2.

GCLUSIM splits the information of these authors along the non-overlapping communities if its abstract feature vector similarity is < 0.2. 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 are retrieved and saved in a list (lines 6–7). If feature vector similarity is less than the threshold, then for each such homonym the intersection of hub publications and community publications is found. A new node is created in the co-authorship graph for each community detected that has the same name as that of the hub node and has an identity one more than in the current nodes in the co-authors graph, this newly created node has the publication list that is found at line 12. Likewise, for all detected communities that have feature vector similarity less than the threshold new nodes in the co-authors

Algorithm 1: Homonyms Resolver Algorithm

```
   Data: Co − authors Graph(CG)
   Result: Homonym Resolved Graph(HG)
 1 [CommunitiesList, Hub, Outliers] ⟵ runSCAN(CG)
 2 HubPubs ⟵ getPubs(Hub)
 3 FV_h ⟵ getFV(Hub)
 4 i, j ⟵ ∅
 5 for community ∈ CommunitiesList do
 6     communityPubs ⟵ ∅
 7     FV_c ⟵ getFV(Community)
 8     FV.similarity ⟵ sim(FV_C, FV_h) if
       FV.similarity < 0.2 then
 9         for node ∈ community do
10             communityPubs ⟵
               communityPubs ∪ getPubs(node)
11         end
12         HN_iPubs ⟵ HubPubs ∩ communityPubs
13         HN_iName ⟵ getName(Hub)
14         HN_iId ⟵ getLength(CG)
15         CG.InsertNode(HN_iId, HN_iName, HN_iPubs)
16         UpdateGraph(CG, community)
17         HubPubs ⟵ HubPubs \ communityPubs
18         i ⟵ i + 1
19     else
20         community ⟵ community + 1
21     end
22 end
23 for outlier ∈ outliers do
24     outlierPublications ⟵ ∅
25     FV_O ⟵ getFV(outlier)
26     FV.similarity ⟵ sim(FV_O, FV_h)
27     if FV.similarity < 0.2 then
28         O_jPubs ⟵ HubPubs ∩ outlierPubs
29         O_jName ⟵ getName(Hub)
30         O_jId ⟵ getLength(CG)
31         CG.InsertNode(O_jId, O_jName, O_jPubs)
32         UpdateGraph(CG, outlier)
33         HubPubs ⟵ HubPubs \ outlierPubs
34         j ⟵ j + 1
35         returnUpdatedGraphCG
36     else
37         outlier ⟵ outlier + 1
38     end
39 end
```

Algorithm 2: Feature Vector Constructor Algorithm

```
   Data: Clusters of Nodes
   Result: FeatureVector(FV)
 1 Inititlize FV ⟵ 0
 2 while Clusters do
 3     text ⟵ get.Abstract(Cluster)
 4     tokens ⟵ get.Tokens(text)
 5     for token ∈ tokens do
 6         if token ∉ FV then
 7             FV.Add(token)
 8             token ⟵ token + 1
 9         else
10             token ⟵ token + 1
11         end
12     end
13 end
14
```

Author's personal copy

graph are created and the graph is updated after every new node insertion. Publications of hub node are also updated accordingly. In graph update, some edges to the hub node to existing communities are removed and some new edges to newly created nodes are created (lines 12–22). The same procedure is repeated for all outliers that have feature vector similarity < 0.2, as done in the case of all communities (lines 23–39). This whole process is shown in Fig. 5.

Here hub node A is split into three new nodes ($A1$, $A2$, $A3$), as there are only three clusters of nodes (communities) that have feature vector similarity < 0.2. Cluster (P, B, C, D) and cluster (F, G) have feature vector similarity > 0.2, so they are not split. GCLUSIM's this feature is very useful for the detection of the same authors (like Ph.D. advisors) that have no overlapping between some of their co-authors, as many Ph.D. students work with their advisors as co-authors and then go to their home coun-

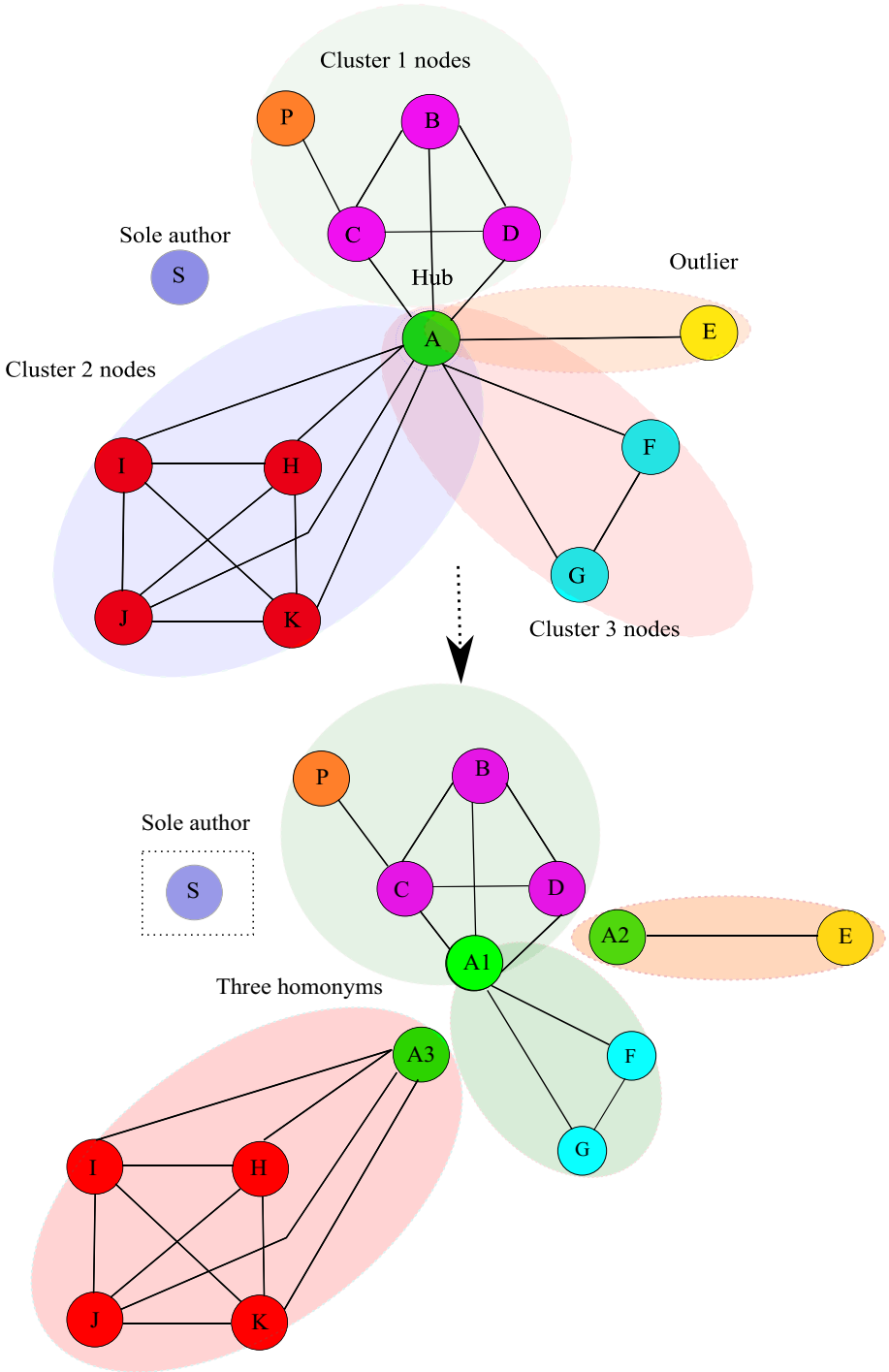


Fig. 5 An example of homonym resolution

Author's personal copy

try and never co-authors again with them. In this way, the homonym problem is solved using graph structural clustering and graph operations. The use of SCAN here brings three benefits (1) It detects outliers that are not possible with existing graph-based methods. (2) It makes algorithm scalable as its computational complexity is much lower than other competing graph methods, as given in Sect. 1. (3) In contrast to other community detection algorithms, it is useful to identify overlapping communities in the co-authors graph

3.3 Proposed Synonym Resolver Algorithm

The other name ambiguity is the synonym that occurs when an author writes her name in different ways as explained in the following paragraphs. In academic domain, authors often fill out their names in several forms like abbreviations, sequence changing, an omission of some part of their name, etc. So, the same author may appear in DL with different forms of her name.

3.3.1 Proposed Hybrid Similarity Index

The synonyms are identified with the help of proposed hybrid similarity index that is given in Eq. 5; hereafter, we simply call this as hybrid similarity. We used Jaro-Winkler similarity for finding similarity between two author name strings. This measure proved to be useful for short strings such as names [37]. The similarity score is normalized in a range of [0–1], where 0 means no similarity and 1 means the exact same strings. Given two name strings n_1 and n_2, their distance is calculated as

$$d(n_1, n_2) = \frac{1}{3} * \left(\frac{c}{n_1} + \frac{c}{n_2} + \frac{c-m}{c} \right) \tag{4}$$

where c is the number of common characters in two name strings. A character is considered as a common character at position i in the string n_1 has to be within the H window of the equivalent jth character in the string n_2. Here $H = \left\lfloor \frac{\max(|n_1|,|n_2|)}{2} \right\rfloor - 1$. Similarly m is equal to the number of characters matched from the window but not at the same index divided by 2. After computing syntactic similarity, we find the geodesic distance between two compared nodes and pass these values to our proposed hybrid similarity index.

Compatible names are those names that are either fully part of another name or they are having same initial of the first and last name. For example, a name "John Smith" is fully part of another name "John Michael Smith" and a name "John Gold" is not having the same initials as that of "John Smith". Similarly, another pair of names is "J. Smith" and "J. Smith Gold" that has the same initials. So, the first and third pair of names is compatible and the second pair is not compatible.

Definition 1 HYBRID SIMILARITY

Let α is the Jaro-Winkler similarity between two compatible compared names and β is the geodesic distance between them. Then hybrid similarity will be given by

$$\text{Hybrid Similarity} = \frac{1}{2}\left(\alpha + \frac{1}{1+\beta^2}\right) \tag{5}$$

If two names are compatible then we find similarity between them using the formula given in Eq. 5. The co-authors graph indicates the author's past and present connections with other researchers. So, if apparently different researchers in co-authors graph have greater hybrid similarity then it is probably the same authors. In GCLUSIM, the nodes that have hybrid similarity > 0.5 are considered to be the same authors else different ones. GCLUSIM first searches for the nodes that have compatible names and applies hybrid similarity on these name pairs as given in Eq. 5. Pseudocode for computing hybrid similarity and finding similar names is described in Algorithm 3 (lines 7–16).

Algorithm 3: Synonyms Resolver Algorithm

```
   Data: Homonym Resolved Graph(HG)
   Result: Synonym Resolved Graph(SG)
 1 HubName ⟵ HG.getName(Hub)
 2 allAuthorNames, similarNamesList ⟵ ∅
 3 allAuthorNames ⟵ HG.getAllName(node)
 4 comparisonName ⟵ HubName
 5 nameTokens ⟵ comparisonName.split()
 6 allAuthorNames.delete(HubName)
 7 for author ∈ allAuthorNames do
 8     if isAuthorCompatible then
 9         α ⟵
           getJWSimilarity(comparisonName, author)
10         β ⟵
           geodesicDistance(comparisonName, author)
11         hybridSimilarityIndex ⟵
           gethybridSimilarityIndex(α, β)
12         if hybridSimilarityIndex > 0.5 then
13             similarNamesList.append(author)
14         end
15     end
16 end
17 for name ∈ similarNamesList do
18     mPublications ⟵ ∅
19     mPublications ⟵
       comparisonNamePublications ∪ namePublications
20     if nameId < comparisonNameId then
21         mName ⟵ nameName
22         mId ⟵ nameId
23     end
24     mName ⟵ comparisonName
25     mId ⟵ comparisonId
26     HG.InsertNode(mId, mName, mPublications)
27     deleteNode(comparisonNode)
28     UpdateGraph(HG)
29     returnUpdatedGraphHG
30 end
```

Author's personal copy

Two synonym nodes are merged into one node that has publications list of both of these nodes and id of the full name, whereas the other name node is deleted from the graph. Pseudocode for merging similar names is described in Algorithm 3 (lines 17–30).

An example of synonym merging is shown in Fig. 6, since node P and node $A1$ are compatible names and hybrid similarity between them is > 0.5. So in this case, they are merged into one node $A1$ and the other node P is deleted from the co-authors graph and the edges are updated accordingly. The proposed hybrid similarity not only considers the syntactic

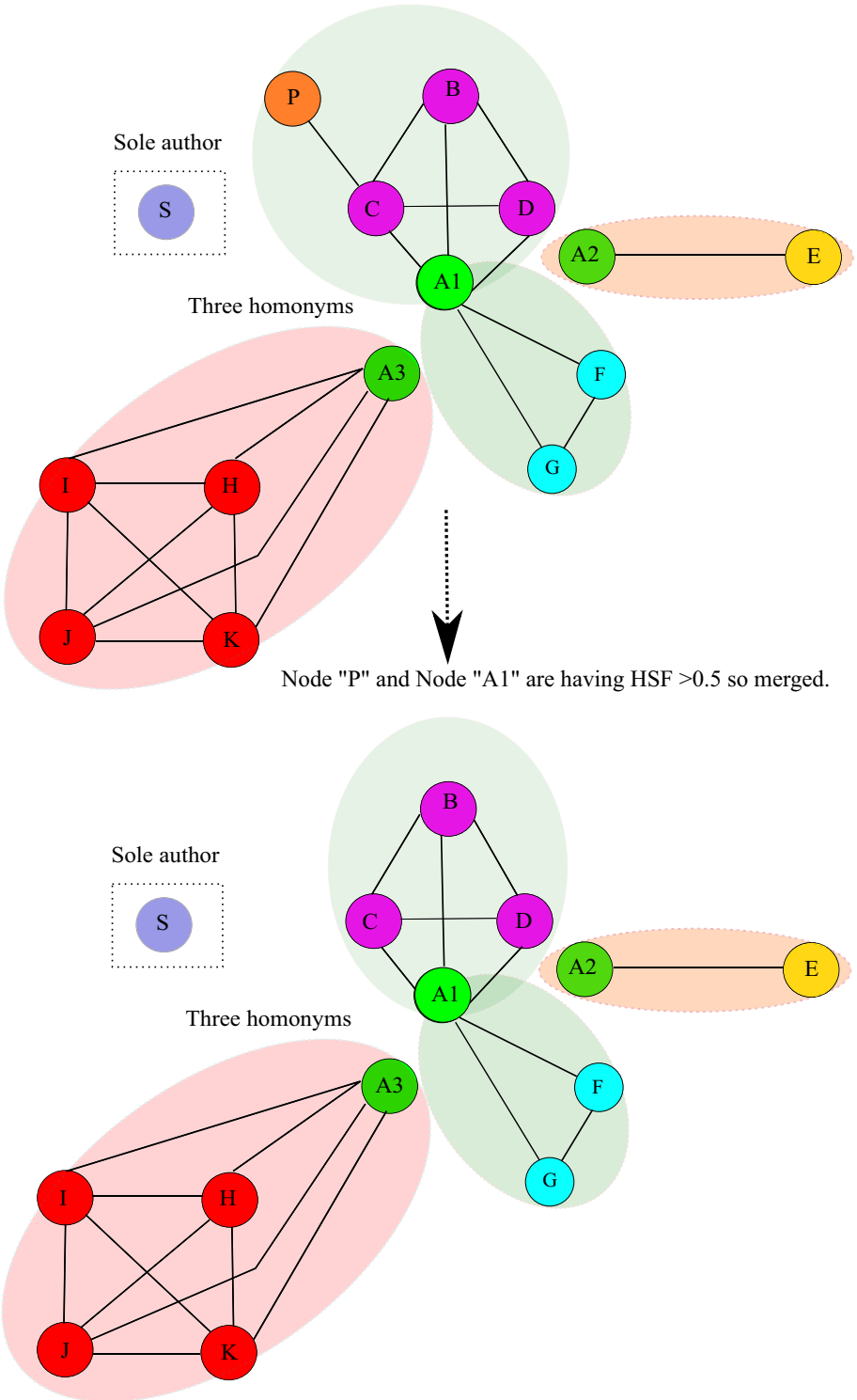


Fig. 6 An example of synonym resolution

Author's personal copy

similarity, but it also takes into account the close relationship of nodes in co-authors graph. This similarity may also be able to detect some typo errors and improves performance.

3.4 Proposed Sole Author Resolver Algorithm

Despite the fact that the resolution of the sole authors is necessary for the completeness of clusters and for the complete solution of AND, the majority of previous studies [2,11,34] ignored the sole author's cases. So, for the sake of cluster completeness and complete solution to the author name ambiguity problem, GCLUSIM finds the nodes that have no co-authors and adds them into the sole author's list. It then takes one by one these sole authors and merges them with the most similar node on the basis of abstract feature vector similarity if it is > 0.2. We used Jaccard index for feature vector similarity because it is computationally fast, as given in Eq. 6. The sole author resolver algorithm proceeds as follows. It finds all sole authors in the graph by selecting those nodes in co-authors graph that has no co-authors (lines 4–8). Then it takes one by one sole authors and finds their similar names in the co-authors graph that have similarity higher than 0.8 to sole author nodes (lines 13–17) and selects the node which has the highest value of similarity between the feature vectors of two nodes provided that it suffice the condition of minimum threshold of 0.2. Finally, sole authors are deleted from the co-authors graph by merging them with the selected node between their abstract feature vectors and update the co-authorship graph (lines 19–34). An example of merging the sole author node to most similar node is shown in Fig. 7, where it is found that sole author node S and similar author node $A1$ have similarity > 0.2. So, these two nodes are merged and graph is updated accordingly.

Algorithm 4: Sole Authors Resolver Algorithm

```
   Data: Synonyms Resolved Graph(SG)
   Result: Disambiguated Graph(DG)
 1 soleAuthorsList ⟵ ∅
 2 allAuthorsList ⟵ ∅
 3 similarAuthorsList ⟵ ∅
 4 for node ∈ SG do
 5     coAuthors ⟵ node.getCoAuthors()
 6     if coAuthors < one then
 7         soleAuthorsList ⟵
           soleAuthorsList ∪ SG.getName(node)
 8     end
 9     allAuthorsList ⟵ SG.getAllNames(node)
10     for author ∈ soleAuthorsList do
11         for comparisonName ∈ allAuthorsList do
12             similarity ⟵
               nameSimilarity(comparisonName, author)
13             if similarity > threshold and
               authorCompatible then
14                 similarAuthorsList.append(author)
15             end
16         end
17     end
18 end
19 for soleAuthor ∈ soleAuthorsList do
20     sFV ⟵ CG.getFV(soleAuthor)
21     maxNode, maxValue ⟵ null
22     for author ∈ similarAuthorsList do
23         aFV ⟵ CG.getFV(author)
24         FV.similarity ⟵ sim(sFV, aFV)
25         if FV.similarity > 0.2 then
26             mergNode ⟵ author
27             maxValue ⟵ FV.similarity
28             maxNode ⟵ soleAuthor
29         else
30             soleAuthor ⟵ soleAuthor + 1
31         end
32         returnUpdatedGraphSG
33     end
34 end
```

4 Experiments and Discussions

In this section, the performance of GCLUSIM is compared with three baseline methods using Arnetminer and BDBComp data set in the form of clustering evaluation metrics.

4.1 Data Set: Arnetminer and BDBComp

Two real-world data sets from Arnetminer and BDBComp are used to measure the effectiveness of GCLUSIM. The statistics of the Arnetminer data set are given in Table 2 and more details are found online at Arnetminer. The original version of this collection was created by [16]. Then, [3] manually checked, labeled and included more ambiguous authors to expand this data set. It is used with slight variations in many AND studies [3,5,25]. Subsets of this data set have also been used in other works [2,4,6,8,10,19].

BDBComp is relatively a small data set of 363 records belonging to 184 distinct authors, but it is very difficult to disambiguate as some authors have only one citation record. BDBComp data set statistics are given in Table 3, and this collection has also been used in many author name disambiguation tasks.

In GCLUSIM, only co-authors information and abstract of this data set are used for the complete solution of the author name ambiguity problem. GCLUSIM considered two authors same if their full name is identical in citation data set. Abstract keyword feature vectors are generated for all publications in the data set. The Jaccard index

$$\text{Jaccard}(A, B) = \frac{A \cap B}{A \cup B} \tag{6}$$

between these feature vectors is used for resolving homonyms and sole authors.

Author's personal copy

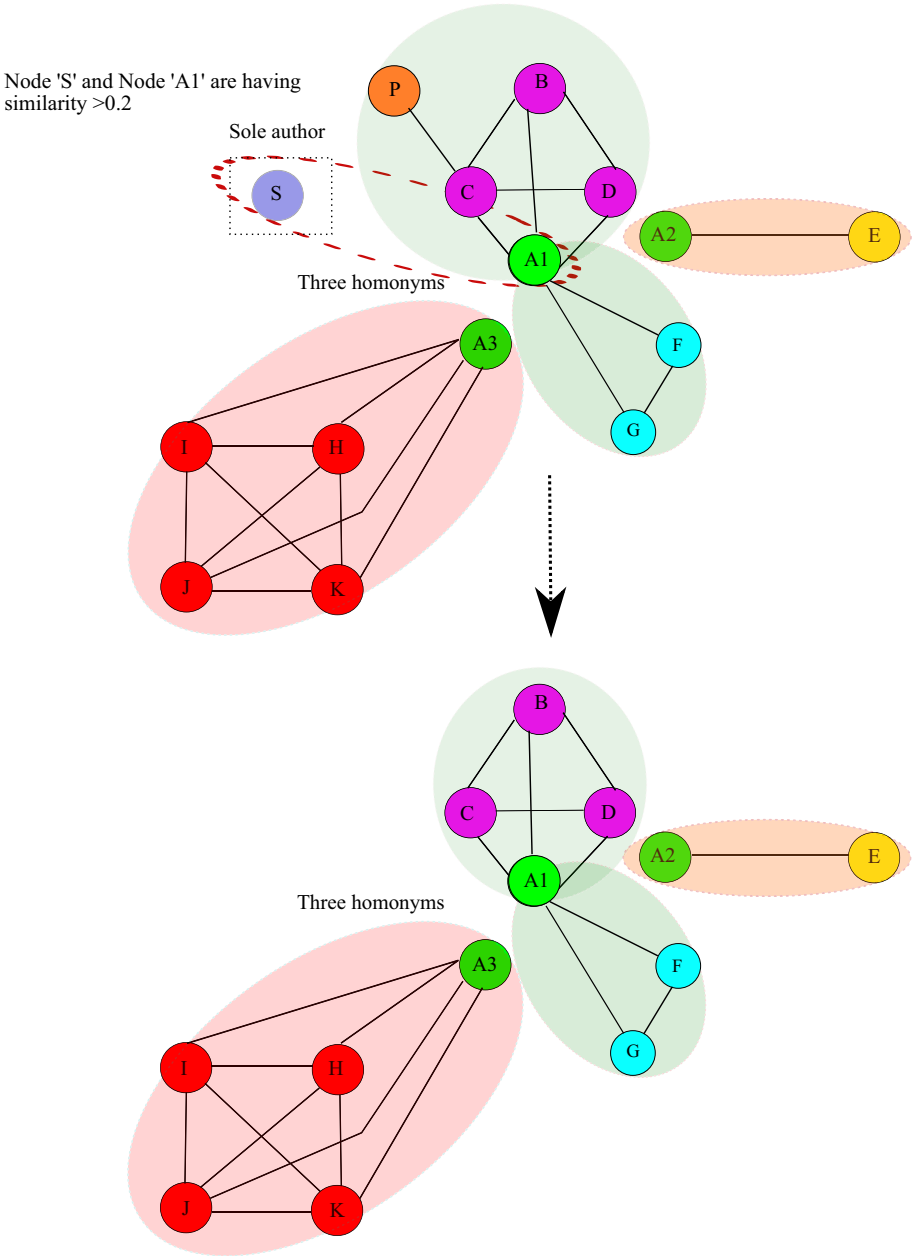


Fig. 7 An example of sole authors resolution

4.2 Evaluation Metrics

Three metrics—K-metric, pairwise-F1, and cluster-F1—are used to measure the effectiveness of GCLUSIM, as defined in [38].

K-metric is defined as the geometric mean of the Average Cluster Purity (ACP) and the Average Author Purity (AAP). ACP evaluates the purity of the algorithm-generated clusters with respect to the gold standard reference clusters, so it measures the amount of data misclassified into the wrong clusters by checking whether the generated clusters include only the publication records belonging to the reference clusters. AAP evaluates the level of splitting author information into several clusters, so it checks how fragmented the generated clusters are. ACP, AAP and K-metric are expressed in Eq. 7.

$$\begin{aligned} \text{ACP} &= \frac{1}{N}\sum_{r=1}^{R}\sum_{s=1}^{S}\frac{n_{rs}^2}{n_r}, \text{AAP} = \frac{1}{N}\sum_{s=1}^{S}\sum_{r=1}^{R}\frac{n_{rs}^2}{n_s}, \\ K &= \sqrt{\text{ACP} * \text{AAP}} \end{aligned} \tag{7}$$

Here, N denotes the size of the citations in the collection, S is the number of gold standard reference clusters manually generated, and R is the number of clusters automatically generated by the GCLUSIM. Also, n_s is the number of elements in cluster s and n_{rs} is the number of elements belonging to

Author's personal copy

Table 2 The Arnetminer data set (Authors denotes the number of distinct authors and Records represents citation records associated with that author)

Name	Authors	Record
Ajay Gupta	11	93
Bin Zhu	15	49
Charles Smith	4	7
Michael Siege	6	54
David C. Wilson	5	67
Eric Martin	5	85
Yu Zhang	72	236
Sanjay Jain	4	216
Hui Yu	21	32
Xiaoyan Li	6	33
Jie Yu	9	32
Bo Liu	65	306
Kai Zhang	28	82
Bin Li	65	306
Michael Wagner	14	71
Paul Brown	7	26
Peter Phillips	3	13
Lu Liu	17	58
Cheng Chang	5	27
David Brown	25	61
David Cooper	7	18
R. Ramesh	9	46
Fei Su	4	37
Gang Luo	9	47
Hui Fang	8	42
Jie Tang	6	66
Yang Yu	19	71
John F. McDonald	2	34
Lei Wang	109	307
Lei Fang	7	17
Ping Zhou	18	36
Rakesh Kumar	10	96
Shu Lin	2	76
Thomas D. Taylor	3	4
Mark Davis	6	24
Michael Lang	4	17
Lei Chen	36	192
Ning Zhang	31	125
Paul Wang	7	16
Robert Allen	9	24
S Huang	13	14
J. Guo	10	13
Ji Zhang	16	64
Wen Jao	9	487
X. Zhang	40	62
Yan Tang	6	27
Wei Xu	47	153
Xiaoming Wang	14	41
Lei Jin	6	16
Li Shen	6	65

Table 3 The BDBComp data set

S. No.	Ambiguous group	Total records	Distinct authors
1	A. Oliveira	52	16
2	A. Silva	64	32
3	F. Silva	26	20
4	J. Oliveira	48	18
5	J. Silva	36	17
6	J. Souza	35	11
7	L. Silva	33	18
8	M. Silva	21	16
9	R. Santos	20	16
10	R. Silva	28	20

both cluster r and cluster s. Pairwise F1 (PF1) is defined as the harmonic mean of Pairwise Precision (PP) and Pairwise Recall (PR). PP is the fraction of publication records corresponding to the same author in the algorithm-generated clusters, and PR is the fraction of publication records associated with the same author in the gold standard reference clusters. The PP, PR and PF1 measures are expressed in Eq. 8, where $C(n, r)$ denotes the number of combinations of r elements from n elements: $C(n, r) = \frac{n!}{r!(n-r)!}, n \geq r$. Other parameters including r, s, n_s and n_{sr} are defined as before in Eq. 7.

$$\begin{aligned} \text{PP} &= \frac{\sum_{r=1}^{R}\sum_{s=1}^{S} C(n_{rs}, 2)}{\sum_{r=1}^{R} C(n_r, 2)}, \\ \text{PR} &= \frac{\sum_{r=1}^{R}\sum_{s=1}^{S} C(n_{rs}, 2)}{\sum_{r=1}^{S} C(n_s, 2)}, \quad \text{PF1} = \frac{2 * \text{PP} * \text{PR}}{\text{PP} + \text{PR}} \end{aligned} \tag{8}$$

Cluster F1 (CF1) is defined as the harmonic mean of Cluster Precision (CP) and Cluster Recall (CR), where CP is the fraction of the generated clusters that are equal to the reference clusters and CR is the fraction of correctly retrieved clusters from the reference clusters. The CP, CR and CF1 measures are given in Eq. 9.

$$\text{CP} = \frac{R \cap S}{R}, \text{CR} = \frac{R \cap S}{S}, \text{CF} - 1 = \frac{2 * \text{CP} * \text{CR}}{\text{CP} + \text{CR}} \tag{9}$$

Author's personal copy

4.3 Baseline Methods

Three graph-based author name disambiguation methods—On Graph-based Name Disambiguation (GHOST) [12], scalable clustering methods (MGPM) [23] and Author name disambiguation using a graph model (GFAD) [5]—were chosen as baseline methods to compare with GCLUSIM.

GHOST was proposed by [12], which they called "GrapHical framewOrk for name diSambiguaTion (GHOST)". It modeled the relationships among publications using undirected graphs and solved name disambiguation 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. MGM and MGPM repeatedly merged the sub-graphs until the number of sub-graphs is equal to the given number of clusters k in the gold standard. MGPM evaluated on the same Arnetminer data set to make a comparison with GCLUSIM. More details of the MGPM method can be seen in [23]. GFAD as proposed by [5] resolves homonyms by finding maximum non-overlapping cycles in co-authors graph. It then solves synonyms by applying longest common subsequence with a threshold of 0.8 and find a connection between two nodes in co-authors graph. In the last step, it solves sole author by merging the sole authors with the most similar author node with the help of title similarity.

4.4 Performance Evaluation of GCLUSIM

GCLUSIM effectiveness is measured using nine clustering evaluation metrics on Arnetminer collection of citation records. Figure 8 shows average ACP, AAP, K-metric, PP, PR, PF1, CP, CR and CF1 of GCLUSIM on Arnetminer data set. On average it achieves both ACP and PP of 97% , K and PF1 are 86% and 80%, respectively, and CF1 of 58%.

Table 4 lists the complete details of each ambiguous author group in the Arnetminer collection. In general ACP and PP of GCLUSIM in the majority of ambiguous author cases are nearly equal to 100% but in some cases such as X. Zhang and Paul Brown it is low. CF1 is sometimes as low as 0%. This usually happens when there are few imbalance clusters (one of them is having less than 10% and the other having more than 90% citations). This happened in the case of John F. McDonald where there are two ground truth clusters, one cluster has only one citation and the other one has 33 citations in it.

Average ACP, AAP, K-metric, PP, PR, PF1, CP, CR and CF1 of GCLUSIM on BDBComp data set are shown in Fig. 9. As seen in this figure, overall results are comparable to the results of Arnetminer collection. However, the average K-metric was dropped to 3.5%. This might be due to a large number of authors have only one citation, so these are wrongly merged with others. On the other hand, GCUSIM

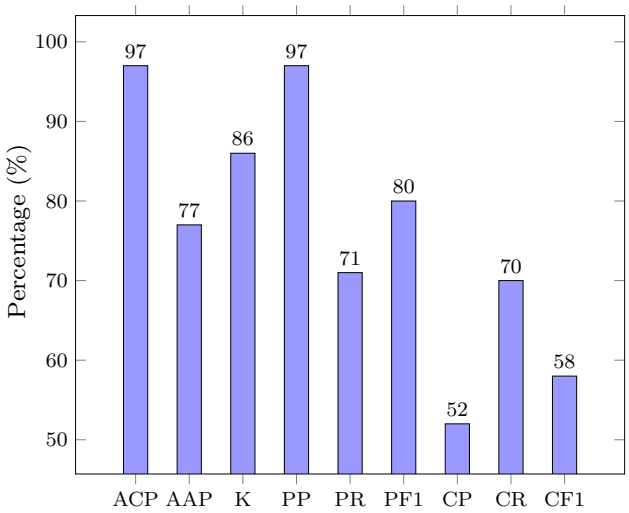


Fig. 8 Average ACP, AAP, K-metric, pairwise precision, pairwise recall, pairwise F1, cluster precision, cluster recall and cluster F1 of the GCLUSIM on Arnetminer

achieved PF1 and CF1 of 84% and CF1 of 62% that is higher as compared to Arnetminer.

In Fig. 10, GCLUSIM performance is compared with baseline methods—GFAD, GHOST, and MGPM. GCLUSIM achieves 13, 10 and 18% higher in K-metric and 28.9, 18.4 and 52.6% higher in CF1 to GFAD, GHOST, and MGPM, respectively. GCLUSIM has a loss of 1.25% in PF1 than GFAD, but higher PF1 than GHOST and MGPM. So, its performance is overall better than baselines.

The reason why GCLUSIM has statistically better results than GFAD, GHOST and MGPM is that it exploited graph structural clustering and relationships between authors using hybrid similarity index. As GHOST only uses syntactic similarity, so some wrong merges of the publications on the basis of this similarity may occur. One other reason of its low performance might be its optimization parameters, as these parameters require careful selection and it may not fit equally well to all ambiguous author groups.

In Fig. 11, performance of GCLUSIM is compared to baselines using the BDBComp collection and once again GCLUSIM shows overall better results than these methods. GCLUSIM achieves 6, 3 and 5% higher in K-metric and 5, 16 and 31% higher in CF1 to GFAD, GHOST and MGPM, respectively. GCLUSIM has comparable performance in PF1 than GFAD, but higher than GHOST and MGPM.

GFAD is unable to identify those authors that only share one co-author, whereas GCLUSIM is able to detect these authors. One other possible reason for GFAD comparatively low performance in K and CF1 may be due to its use of only title words for similarity calculations that are not a representative of an author's work. GCLUSIM shows comparable PF1 as the GFAD that outperforms other techniques [4,10,24]. As seen in Figs. 10 and 11, the MGPM perfor-

Author's personal copy

Table 4 The performance evaluation of GCLUSIM for each ambiguous group on the Arnetminer collection

Name	ACP	AAP	K	PP	PR	PF1	CP	CR	CF1
Ajay Gupta	0.87	0.58	0.71	0.80	0.53	0.64	0.19	0.33	0.24
Bin Li	0.95	0.84	0.89	0.93	0.77	0.84	0.72	0.83	0.78
Bin Zhu	1.00	0.86	0.93	1.00	0.73	0.84	0.68	0.87	0.76
Charles Smith	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
David C. Wilson	1.00	0.40	0.63	1.00	0.33	0.50	0.25	0.60	0.35
Eric Martin	0.99	0.56	0.74	1.00	0.56	0.71	0.18	0.60	0.27
Fei Su	0.96	0.71	0.83	0.99	0.76	0.86	0.25	0.50	0.33
Hui Yu	1.00	0.97	0.98	1.00	0.95	0.97	0.91	0.95	0.93
Ji Zhang	0.98	0.83	0.90	0.99	0.84	0.91	0.50	0.69	0.58
Jie Yu	1.00	0.89	0.94	1.00	0.97	0.99	0.50	0.67	0.57
Kai Zhang	0.98	0.84	0.91	0.99	0.82	0.90	0.55	0.71	0.62
Bo Liu	0.97	0.80	0.88	0.91	0.47	0.62	0.78	0.89	0.83
Cheng Chang	0.96	0.67	0.81	0.98	0.53	0.69	0.33	0.60	0.43
David Brown	0.98	0.84	0.91	0.99	0.83	0.90	0.71	0.88	0.79
David Cooper	1.00	0.92	0.96	1.00	0.90	0.95	0.75	0.86	0.80
Gang Luo	0.96	0.85	0.90	0.99	0.76	0.86	0.60	0.67	0.63
Hui Fang	1.00	0.62	0.79	1.00	0.52	0.69	0.29	0.50	0.36
J. Guo	1.00	0.92	0.96	1.00	0.67	0.80	0.82	0.90	0.86
Jie Tang	1.00	0.73	0.85	1.00	0.69	0.82	0.38	0.83	0.53
John F. McDonald	0.91	0.94	0.93	0.94	0.94	0.94	0.00	0.00	0.00
Lei Chen	0.97	0.65	0.79	0.95	0.49	0.64	0.58	0.80	0.67
Lei Fang	1.00	0.90	0.95	1.00	0.77	0.87	0.75	0.86	0.80
Lei Wang	0.93	0.84	0.88	0.85	0.71	0.77	0.71	0.82	0.76
Lu Liu	0.97	0.84	0.90	0.99	0.73	0.84	0.65	0.88	0.75
Michael Lang	1.00	0.72	0.85	1.00	0.54	0.70	0.50	0.75	0.60
Ning Zhang	0.98	0.56	0.74	1.00	0.30	0.46	0.43	0.70	0.53
Paul Wang	0.94	0.92	0.93	0.95	0.90	0.93	0.71	0.71	0.71
S Huang	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Wen Jao	0.99	0.89	0.94	0.99	0.90	0.94	0.24	0.70	0.36
X. Zhang	0.79	0.90	0.84	0.47	0.76	0.58	0.67	0.60	0.63
Xiaoyan Li	0.95	0.90	0.92	0.94	0.84	0.88	0.57	0.67	0.62
Yan Tang	1.00	0.85	0.92	1.00	0.81	0.90	0.64	0.82	0.72
Lei Jin	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Li Shen	0.99	0.70	0.83	1.00	0.69	0.82	0.33	0.67	0.44
Mark Davis	1.00	0.57	0.75	1.00	0.34	0.51	0.40	0.67	0.50
Michael Siege	0.96	0.75	0.85	1.00	0.77	0.87	0.25	0.50	0.33
Michael Wagner	0.96	0.51	0.70	0.95	0.26	0.41	0.29	0.50	0.37
Paul Brown	0.86	0.77	0.81	0.85	0.66	0.75	0.56	0.62	0.59
Peter Phillips	1.00	0.77	0.88	1.00	0.74	0.85	0.20	0.33	0.25
Ping Zhou	1.00	0.92	0.96	1.00	0.82	0.90	0.80	0.89	0.84
R. Ramesh	1.00	0.54	0.74	1.00	0.48	0.65	0.32	0.78	0.45
Rakesh Kumar	0.99	0.66	0.81	1.00	0.74	0.85	0.19	0.50	0.28
Robert Allen	0.96	0.92	0.94	0.99	0.87	0.92	0.78	0.78	0.78
Sanjay Jain	0.99	0.62	0.78	1.00	0.67	0.80	0.04	0.20	0.06
Shu Lin	1.00	0.38	0.62	1.00	0.37	0.54	0.08	0.50	0.14
Thomas D. Taylor	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Wei Xu	0.96	0.72	0.83	0.94	0.63	0.76	0.47	0.65	0.54
Xiaoming Wang	0.97	0.80	0.88	1.00	0.89	0.94	0.50	0.64	0.56

Author's personal copy

Table 4 continued

Name	ACP	AAP	K	PP	PR	PF1	CP	CR	CF1
Yang Yu	0.99	0.72	0.84	0.99	0.57	0.72	0.47	0.74	0.57
Yu Zhang	0.95	0.65	0.78	0.91	0.44	0.59	0.49	0.69	0.57

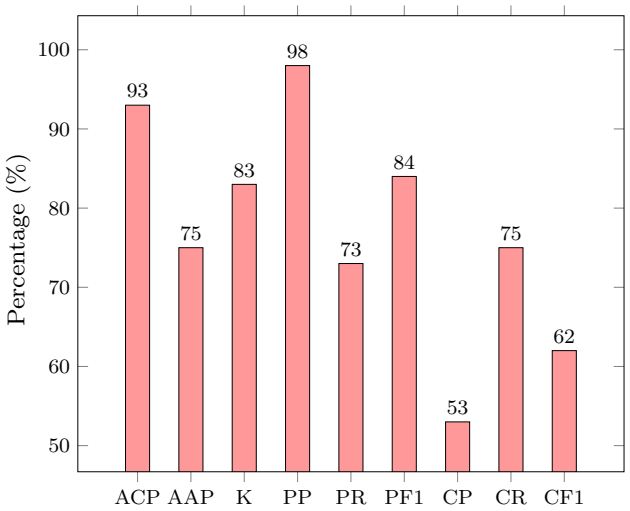


Fig. 9 Average ACP, AAP, *K*-metric, pairwise precision, pairwise recall, pairwise F1, cluster precision, cluster recall and cluster F1 of the GCLUSIM on BDBComp

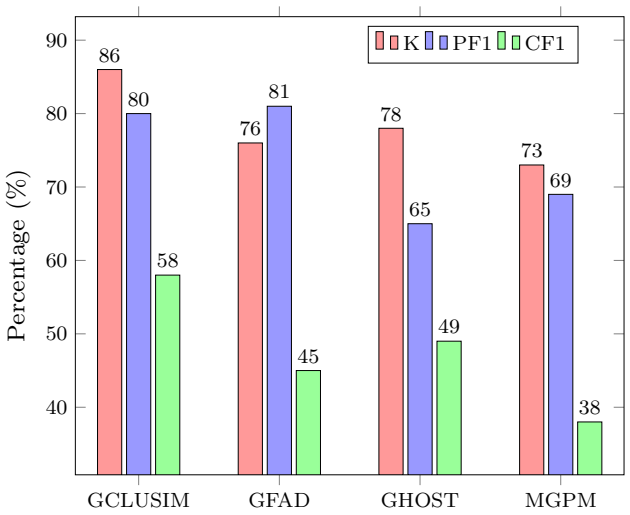


Fig. 10 Average K, pairwise F1 and cluster F1 of the GCLUSIM, GFAD, GHOST and MGPM on Arnetminer

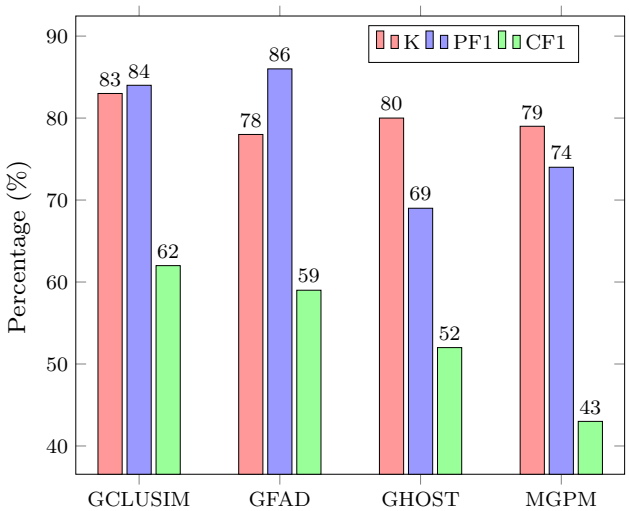


Fig. 11 Average K, pairwise F1 and cluster F1 of the GCLUSIM, GFAD, GHOST and MGPM on BDBComp

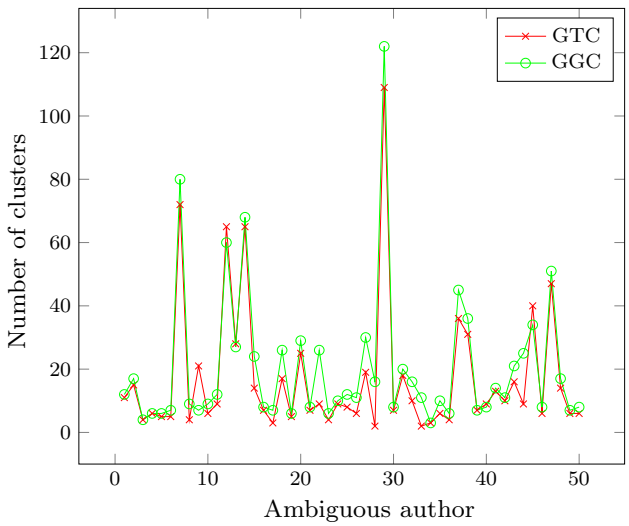


Fig. 12 Comparison of ground truth clusters (GTC) and GCLUSIM generated clusters (GGC)

mance is overall worst as compared to GCLUSIM, even though it used the hidden information about the exact number '*K*' of ambiguous author groups. As in Arnetminer 4.3% authors are sole authors and 35.6% have only one citation, so MGPM might perform some wrong partitioning and merging of these authors in both phases.

Comparison between ground truth clusters and GCLUSIM generated clusters is shown in Fig. 12. About 80% clusters are within the range (Ground Truth Clusters ± 5) that's why GCLUSIM shows better CF1 than baseline methods. However, GCLUSIM produced more clusters than the ground truth clusters. This is due to the fact that few authors have changed their research interests and thus have different social circles.

Author's personal copy

Table 5 Summary of running times (seconds) on Arnetminer and BDBComp data set of all methods

Data set	GHOST	MGPM	GFAD	GCLUSIM
Arnetminer	57	49	749	38
BDBComp	31	24	257	29

4.5 Run Time Analysis of GCLUSIM

The GCLUSIM consists of five stages, in which graph structural clustering is rather time-consuming step as compared to other steps. To show the scalability of GCLUSIM, the comparison of running time of all methods for disambiguating Arnetminer and BDBComp data sets is shown in Table 5. All experiments were performed on a personal computer with Intel(R) Core(TM)i5-5200U CPU @ 2.20 GHz 2.20 GHz and 8 Gigabyte memory. All methods were implemented using Python. In GHOST and MGPM, we set the threshold parameter 'K' equal to the number of clusters in the gold standard data set for hierarchical clustering. GFAD has two parameters; the first is the maximum non-overlapping cycles in co-authors graph and the other is a threshold for string comparison (longest common subsequence) that we set 0.8. Similarly, in GCLUSIM there are three threshold parameters; the first is 0.8 for optimal blocking; the second is 0.2 for feature vector similarity; and the last one is the 0.5 for the proposed hybrid similarity. After different experiments, we carefully chose these parameters.

As seen in Table 5, GFAD was the slowest among all the methods on both collections. Its most time-consuming stage is the cycle finding in the co-authors graph. GHOST and GCLUSIM were the fastest on BDBComp and Arnetminer, respectively. The reason why GHOST outperforms on BDBComp is that there are small length paths in the co-authors graph of this collection, as the majority of authors have one or two citations. The running time of GLUSIM was statistically tied with GHOST on BDBComp collection. GCLUSIM is about 8–19 times faster than GFAD and overall faster than all other baseline methods.

5 Conclusions and Future work

Semantic similarity and community detection algorithms are little used in the domain of author name disambiguation. In this paper, we have proposed GCLUSIM a graph-based method that does not require costly training data or a priori hidden information (how many ambiguous authors k in an ambiguous block) and web searches, solves both homonyms and synonyms. In contrast to many previous studies, GCLUSIM also solves the sole authors. In this work, we present three algorithms, namely homonyms resolver algorithm, synonyms resolver algorithm and sole authors resolver algorithm, as our main contributions. GCLUSIM performance is tested on Arnetminer and BDBComp. It shows overall better results than baseline methods. GCLUSIM delivers an effective solution to author name ambiguity problem as it utilizes the powers of graph algorithms. However, it is unable to detect very ambiguous author cases. In future, we plan to analyze self-citation and email address of authors to overcome this limitation.

References

1. Bhattacharya, I.; Getoor, L.: Collective entity resolution in relational data. ACM Trans. Knowl. Discov. Data (TKDD) **1**(1), 5 (2007)
2. Ferreira, A.A.; Veloso, A.; Gonçalves, M.A.; Laender, A.H.: Effective self-training author name disambiguation in scholarly digital libraries. In: Proceedings of the 10th Annual Joint Conference on Digital Libraries, pp. 39–48. ACM (2010)
3. Tang, J.; Fong, A.C.; Wang, B.; Zhang, J.: A unified probabilistic framework for name disambiguation in digital library. IEEE Trans. Knowl. Data Eng. **24**(6), 975–987 (2012)
4. Han, H.; Xu, W.; Zha, H.; Giles, C.L.: A hierarchical naive bayes mixture model for name disambiguation in author citations. In: Proceedings of the 2005 ACM symposium on Applied computing, pp. 1065–1069. ACM (2005)
5. Shin, D.; Kim, T.; Choi, J.; Kim, J.: Author name disambiguation using a graph model with node splitting and merging based on bibliographic information. Scientometrics **100**(1), 15–50 (2014)
6. Han, D.; Liu, S.; Hu, Y.; Wang, B.; Sun, Y.: Elm-based name disambiguation in bibliography. World Wide Web **18**(2), 253–263 (2015)
7. On, B.W.; Lee, D.; Kang, J.; Mitra, P.: Comparative study of name disambiguation problem using a scalable blocking-based framework. In: Proceedings of the 5th ACM/IEEE-CS Joint Conference on Digital Libraries, pp. 344–353. ACM (2005)
8. Huang, J.; Ertekin, S.; Giles, C.L.: Efficient name disambiguation for large-scale databases. In: European Conference on Principles of Data Mining and Knowledge Discovery, pp. 536–544. Springer (2006)
9. Treeratpituk, P.; Giles, C.L.: Disambiguating authors in academic publications using random forests. In: Proceedings of the 9th ACM/IEEE-CS Joint Conference on Digital Libraries, pp. 39–48. ACM (2009)
10. Cota, R.G.; Ferreira, A.A.; Nascimento, C.; Gonçalves, M.A.; Laender, A.H.: An unsupervised heuristic-based hierarchical method for name disambiguation in bibliographic citations. J. Am. Soc. Inf. Sci. Technol. **61**(9), 1853–1870 (2010)
11. de Carvalho, A.P.; Ferreira, A.A.; Laender, A.H.; Gonçalves, M.A.: Incremental unsupervised name disambiguation in cleaned digital libraries. J. Inf. Data Manag. **2**(3), 289 (2011)
12. Fan, X.; Wang, J.; Pu, X.; Zhou, L.; Lv, B.: On graph-based name disambiguation. J. Data Inf. Qual. (JDIQ) **2**(2), 10 (2011)
13. Onodera, N.; Iwasawa, M.; Midorikawa, N.; Yoshikane, F.; Amano, K.; Ootani, Y.; Kodama, T.; Kiyama, Y.; Tsunoda, H.; Yamazaki, S.: A method for eliminating articles by homonymous authors from the large number of articles retrieved by author search. J. Am. Soc. Inf. Sci. Technol. **62**(4), 677–690 (2011)
14. Huynh, T.; Hoang, K.; Do, T.; Huynh, D.: Vietnamese author name disambiguation for integrating publications from heterogeneous sources. In: Asian Conference on Intelligent Information and Database Systems, pp. 226–235. Springer (2013)

Author's personal copy

15. Liu, Y.; Tang, Y.: Network based framework for author name disambiguation applications. Int. J. u and e Serv. Sci. Technol. **8**(9), 75–82 (2015)
16. Wang, X.; Tang, J.; Cheng, H.; Philip, S.Y.: Adana: Active name disambiguation. In: 2011 IEEE 11th International Conference on Data Mining, pp. 794–803. IEEE (2011)
17. On, B.W.; Elmacioglu, E.; Lee, D.; Kang, J.; Pei, J.: Improving grouped-entity resolution using quasi-cliques. In: Sixth International Conference on Data Mining (ICDM'06), pp. 1008–1015. IEEE (2006)
18. Peng, H.T.; Lu, C.Y.; Hsu, W.; Ho, J.M.: Disambiguating authors in citations on the web and authorship correlations. Expert Syst. Appl. **39**(12), 10521–10532 (2012)
19. Han, H.; Giles, L.; Zha, H.; Li, C.; Tsioutsiouliklis, K.: Two supervised learning approaches for name disambiguation in author citations. In: Proceedings of the 2004 Joint ACM/IEEE Conference on Digital Libraries, 2004, pp. 296–305. IEEE (2004)
20. Wang, J.; Berzins, K.; Hicks, D.; Melkers, J.; Xiao, F.; Pinheiro, D.: A boosted-trees method for name disambiguation. Scientometrics **93**(2), 391–411 (2012)
21. Xu, X.; Yuruk, N.; Feng, Z.; Schweiger, T.A.: Scan: a structural clustering algorithm for networks. In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 824–833. ACM (2007)
22. Johnson, D.B.: Finding all the elementary circuits of a directed graph. SIAM J. Comput. **4**(1), 77–84 (1975)
23. On, B.W.; Lee, I.; Lee, D.: Scalable clustering methods for the name disambiguation problem. Knowl. Inf. Syst. **31**(1), 129–151 (2012)
24. Tran, H.N.; Huynh, T.; Do, T.: Author name disambiguation by using deep neural network. In: Asian Conference on Intelligent Information and Database Systems, pp. 123–132. Springer (2014)
25. Wu, H.; Li, B.; Pei, Y.; He, J.: Unsupervised author disambiguation using dempster-shafer theory. Scientometrics **101**(3), 1955–1972 (2014)
26. Zhu, J.; Yang, Y.; Xie, Q.; Wang, L.; Hassan, S.U.: Robust hybrid name disambiguation framework for large databases. Scientometrics **98**(3), 2255–2274 (2014)
27. Levin, F.H.; Heuser, C.A.: Evaluating the use of social networks in author name disambiguation in digital libraries. J. Inf. Data Manag. **1**(2), 183 (2010)
28. Shoaib, M.; Daud, A.; Khiyal, M.S.H.: Improving similarity measures for publications with special focus on author name disambiguation. Arab. J. Sci. Eng. **40**(6), 1591–1605 (2015)
29. Al-Safadi, L.; Al-Rgebh, D.; AlOhali, W.: A comparison between ontology-based and translation-based semantic search engines for arabic blogs. Arab. J. Sci. Eng. **38**(11), 2985 (2013)
30. Al-Rajebah, N.I.; Al-Khalifa, H.S.: Extracting ontologies from arabic wikipedia: a linguistic approach. Arab. J. Sci. Eng **39**(4), 2749–2771 (2014)
31. Mansouri, D.; Mille, A.; Hamdi-Cherif, A.: Adaptive delivery of trainings using ontologies and case-based reasoning. Arab. J. Sci. Eng. **39**(3), 1849 (2014)
32. Huang, Z.; Zhang, J.; Zhang, B.: Information recommendation between user groups in social networks. Arab. J. Sci. Eng. **40**(5), 1443–1453 (2015)
33. Liu, Q.; Zhou, B.; Li, S.; Li, A.p; Zou, P.; Jia, Y.: Community detection utilizing a novel multi-swarm fruit fly optimization algorithm with hill-climbing strategy. Arab. J. Sci. Eng. **41**(3), 807–828 (2016)
34. Imran, M.; Gillani, S.; Marchese, M.: A real-time heuristic-based unsupervised method for name disambiguation in digital libraries. D Lib. Mag. **19**(9), 1 (2013)
35. Porter, M.F.: An algorithm for suffix stripping. Program **14**(3), 130–137 (1980)
36. Kang, I.S.; Na, S.H.; Lee, S.; Jung, H.; Kim, P.; Sung, W.K.; Lee, J.H.: On co-authorship for author disambiguation. Inf. Process. Manag. **45**(1), 84–97 (2009)
37. Cohen, W.; Ravikumar, P.; Fienberg, S.: A comparison of string metrics for matching names and records. In: Kdd Workshop on Data Cleaning and Object Consolidation, vol. 3, pp. 73–78 (2003)
38. Pereira, D.A.; Ribeiro-Neto, B.; Ziviani, N.; Laender, A.H.; Gonçalves, M.A.; Ferreira, A.A.: Using web information for author name disambiguation. In: Proceedings of the 9th ACM/IEEE-CS Joint Conference on Digital Libraries, pp. 49–58. ACM (2009)