Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2019) 27: 3665 – 3681
© TÜBİTAK
doi:10.3906/elk-1806-132

Incremental author name disambiguation using author profile models and self-citations

Ijaz HUSSAIN[*], **Sohail ASGHAR**
Department of Computer Science, COMSATS University Islamabad, Islamabad, Pakistan

Received: 18.06.2018 • **Accepted/Published Online:** 05.06.2019 • **Final Version:** 18.09.2019

Abstract: Author name ambiguity in bibliographic databases (BDs) such as DBLP is a challenging problem that degrades the information retrieval quality, citation analysis, and proper attribution to the authors. It occurs when several authors have the same name (homonym) or when an author publishes under several name variants (synonym). Traditionally, much research has been conducted to disambiguate whole bibliographic database at once whenever some new citations are added in these BDs. However, it is more time-consuming and discards the manual disambiguation effects (if any). Only a few incremental author name disambiguation methods are proposed but these methods produce fragmented clusters which lower their accuracy. In this paper, a method, called CAND, that uses author profile models and self-citations for incremental author name disambiguation is proposed. CAND introduces name indices that enhance the overall system response by comparing the newly inserted references to the indexed author clusters. Author profile models are generated for the existing authors in BDs which help in disambiguating the newly inserted references. A comparator function is proposed to resolve the incremental author name ambiguity which utilizes the most strong bibliometric features such as coauthor, titles, author profile models, and self-citations. Two real-world data sets, one from Arnetminer and the other from BDBComp, are used to validate CAND's performance. Experimental results show that CAND's performance is overall better than the existing state-of-the-art incremental author name disambiguation methods.

Key words: Incremental author name disambiguation, author profile models, name indices, self-citations, bibliographic databases

1. Introduction

Due to a limited number of names or some popular names, different authors may have the same name and in contrast to this, an author name may be represented in different ways due to different journals/conferences naming conventions. Author name ambiguities can cause wrong attributions and incorrect search results [1–3]. This is quite common in Asian names, particularly in Chinese and Korean. The methods that resolve these author name ambiguities are called author name disambiguation (AND) methods. The increased growth of scientific publications has made the author name ambiguity problem much harder than in the past. Bollen et al. predicted the substantial growth in coming years for the research articles [4]. In 2010, Jinaha estimated that until now 50 million research articles have been published, and on average one article per minute is being published [5].

Hussain et al., in a recent survey of author name disambiguation techniques, pointed out different challenges in AND methods [6]. One of the major challenges among others is an incremental author name

[*]Correspondence: ijazhussain7979@gmail.com

disambiguation. Citation records are constantly generated and inserted into bibliographic databases (BDs) that are already disambiguated. Each of these newly inserted references r_i should be assigned to their respective real authors. The majority of existing BDs rerun the whole disambiguation process upon new insertions of citations which is called batch AND. However, this is infeasible due to three reasons. First, these methods have scalability issues as they again apply disambiguation algorithms on the whole citation records of these BDs. Second, these methods are not feasible for supervised disambiguation methods because they need to retrain the model on each new citation update/insertion. Third and last, they destroy the manual disambiguation effects which are sometimes necessary for fine-tuning of the disambiguation results.

Although, several AND methods have been proposed (see, for instance, [6]), but the vast majority of those methods work for batch AND. To the best of our knowledge, there are only three incremental AND methods [7–9]. These methods resolve only newly inserted articles and are more effective than batch (traditional) AND, and preserve the manual disambiguation effects (if any).

De Carvalho et al. proposed a solution called "incremental unsupervised name disambiguation in cleaned digital libraries (INDi)", which produces very pure clusters on the basis of title and coauthors, but it splits an author's papers into many authors [7]. Esperidiao et al. enhanced the INDi by dropping the assumption that BDs are already cleaned, and proposed five record-selection strategies for newly inserted records in these BDs and found that a newly inserted record closer to the centroid of the existing disambiguated records called CEN is best among these [8]. The fragment comparison method is used by Santana et al. to retrieve the relevant author blocks to the new record [9]. All these methods suffer from fragmentation problem which lowers their accuracy and compares the new reference to existing clusters by comparing it with all the individual references in that clusters which is not feasible for larger clusters.

In this paper, we propose and evaluate a novel incremental AND method which creates less fragmented clusters and improves the accuracy of the method [1]. We used our previously proposed batch AND method [10] as an input to the proposed incremental AND method (CAND), as shown in Figure 1.

In a nutshell, the main contributions of this paper are as follows:

– A blocking-based name index structure that enhances the overall system response is proposed and author profile models are built, which helps in disambiguating the newly inserted citations,

– A comparison function is presented; this function utilizes the most strong bibliometric features such as coauthors, titles, author profile models, and self-citations to effectively fuse the newly inserted reference to the existing cluster/clusters. Self-citations are used for the first time to solve the incremental author name ambiguity problem. CAND exploits author name indices, author profile models, and a comparison function to solve the incremental author name ambiguity, and

– Experiments on two real-world data sets are performed to validate the effectiveness of CAND. Experimental results show that CAND is overall better than the existing incremental AND methods.

The rest of the paper proceeds as follows: Some studies related to CAND are described in Section 2, while CAND architecture is discussed in Section 3. At Section 4, CAND is compared with existing incremental AND methods. Finally, we conclude the paper and discuss some possible future research directions of our work at Section 5.

[1]In terms of clustering metrics (AAP, ACP, and K-metric)

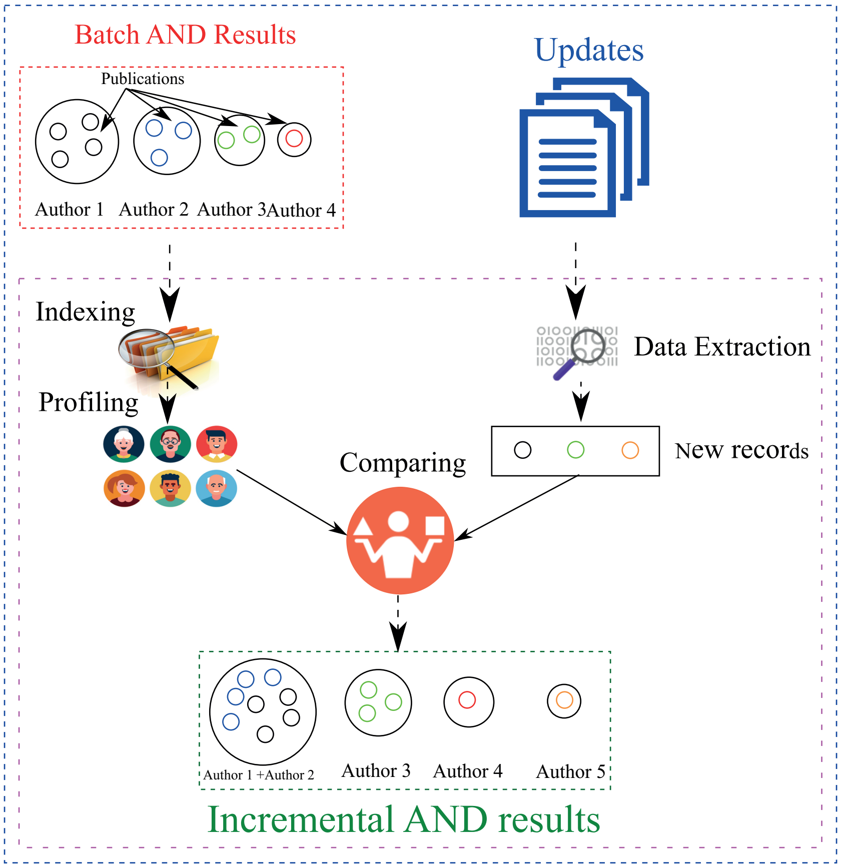

Figure 1. Incremental AND architecture diagram.

2. Related work

Current AND methods can be categorized into two groups: the first are the methods that resolve whole citations in BDs on new insertions called batch AND and the second are those methods which disambiguate only newly inserted citations called incremental AND. Similarly, Hussain et al. in [6] proposed a taxonomy for existing AND methods and divided all methods into supervised [3, 11–13], unsupervised [14–19], semisupervised [20–22], graph-oriented approaches using graph models or social networks [1, 23–25], and string processing or heuristic-based methods [26–28]. In this section, we only overview the incremental AND methods. For detailed discussions about AND methods and AND data sets, interested readers are referred to our recent survey of these techniques presented in [6] and [29], respectively.

Kim et al. proposed an algorithm which consists of two main steps-data generation and matching procedure. In the first step, data that are used for feature matching in the later stage are generated. Per feature matching and clustering is used to generate the training/evaluation data. Experiments are done on the Web of Science data set and the performances are compared [30].

Carvalho et al. proposed INDi a solution for the existing cleaned BDs. INDi utilizes similarity among bibliographic records and groups the new records to authors with similar citation records in the BD or to new authors when the similarity evidence is not strong enough. Heuristics such as titles and coauthors are used for checking whether references of new citation records belong to preexisting authors of the BD or if they belong to new ones (i.e. authors without citation records in the BD), avoid running the disambiguation process on the entire BD. They run simulations on BDBComp collection and synthetic data set to assess the effectiveness of INDi.

Abdulhayoglu and Thijs used Research Gate (RG), connected components, and a graph-based machine learning approach to form groups of authors. Connected components are compared with research gate retrieved author pages for the same author. Then, they retained the groups that have at least 10 authors for detailed examination and employed Google custom search engine API to access the pages of authors for complementing the research gate pages [31].

Esperidiao et al. improved INDi by proposing a new technique INDi+ which try to overcome fragmentation problem that is present in INDi. Of course, these INDi and INDi+ are more efficient than the batch (traditional) AND methods, but they clearly lack a balance between purity and fragmentation of author papers.

Santana et al. proposed "Incremental author name disambiguation by exploiting domain-specific heuristics (INC)" that use fragment comparison method for retrieving the relevant author blocks to the new record [9]. They used author names, coauthor names, titles, and venue for similarity calculation between new records and the retrieved block of records. However, these methods suffer from different problems such as fragmentation, not handling transitivity problem, how to configure their large set of parameters with limited training data.

Zhao et al. presented a naive Bayes probabilistic model which has three stages. In the first stage, they initialize a model and if there is no model present in the existing disambiguation results, and then they use high-precision rules for generating labeled training data. In the second stage, they trained a naive Bayes classifier for each cluster group. In the third stage, clusters that achieve top posterior probabilities are matched with coauthor similarities for predicting its class. This method creates very pure but fragmented clusters [32].

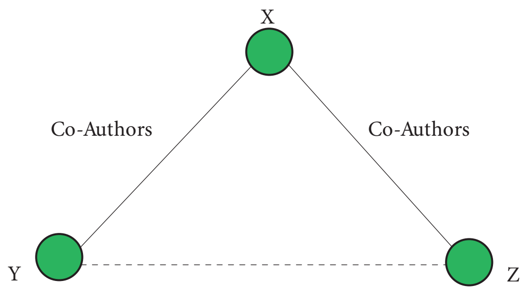

Figure 2. Transitivity problem.

In contrast to existing related methods, CAND uses author profile models, self-citations, and requires no training. CAND compares new ambiguous record to a set of records (clusters) as a whole, so CAND does not suffer from transitivity problem. When there are three or more papers where paper "A" is written by author "X" and author "Y", paper "B" is written by author "X" and author "Z", and another paper is written by author "Y" and author "Z", in the first two papers, we are not sure that author "X" in the first paper and author "X" in the second paper are the same. This is called transitivity problem as there is no direct relationship between these two authors, but indirect relationship via their coauthors as shown in Figure 2. Generally, in AND domain two papers are compared at a time and hence those methods are unable to solve this problem. CAND compares new papers with an existing cluster. Therefore, it does not suffer from transitivity problem. CAND is similar to INDi, INDi+, and INC as these methods use coauthors, and titles for disambiguation. A comparison of CAND with state-of-the-art methods is given in Table 1.

Table 1. A comparison between CAND and other related works (Ref. is Reference, Inc. is Incremental).

Ref.	Inc.	Methodology	Dataset	Evidence used	Capability	Limitations
[30]	No	Triangulation approach	DBLP	Automatic combined with manual labeling	Both	Manual labeling is time-consuming and compared with baseline algorithms.
[31]	No	Research Gate and Google custom search engine were compared with clustering results.	Web of Science Core Collection	Research Gate Profile and Google Scholar pages	Homonyms	External web accesses make it slow.
[32]	Yes	A probabilistic naive Bayes model that simultaneously uses a rich set of metadata and reduces the amount of pairwise comparisons needed for new articles.	Web of Science	Authors, coauthors, e-mail, self-citation, middle initial with subject, exact citation and exact venue.	-	Parameter sweeping is utilized.
[33]	No	High-precision rules are used for generating initial training instances for the training of the supervised method.	QIAN, Arnetminer	ORCIDs linkage, e-mail address, authors, coauthors, citations	Homonyms	Performance of the proposed method relies on the availability of matching features.
[9]	Yes	Fragment comparison method	KISTI, SyGar,BDBComp	Authors, coauthors, e-mail and venue.	Homonyms	Fragmentation, not handling transitivity problem, how to configure their large set of parameters with limited training data
[8]	Yes	AND heuristic such as similar titles and coauthors are used.	BDBComp, SyGar	Titles and coauthors	-	Fragmented clusters.
CAND	Yes	Author profile model combined with heuristics and self-citation.	Arnetminer, BDB-Comp	Authors, coauthors, titles, self-citations, author signatures.	Both	Cannot handle the case of very ambiguous authors.

3. The proposed system: CAND

CAND consists of the following main modules: index creator, authors profile builder, data extractor, and comparator, as shown in Figure 1. There are two inputs to the CAND algorithm. The first one is the disambiguated results using the batch AND, and the second is the incremental updates of publications received from different publishing venues. CAND assumed that at the start the BD is disambiguated. Thus, we use the output of our previously proposed batch AND method as an input to CAND [10]. It is worth to note that the batch AND method is applied only once and in subsequent citation loads, only the CAND algorithm is used for disambiguation.

3.1. Indices creator

Author node consists of node ID, node name, and node publications. All author names are parsed into two components: first name and last name. We index all these names using the last names of the authors. This index is important for efficient retrieval of the results, and it improves the overall performance of the system. The structure of the author name index is shown in Figure 3.

Author names are divided into first name and last name. Indices are created using the last name of the authors, for example, Akram Abay and Altaf Abay would go into the same block “Abay, A” as shown in Figure 3. Any name having more than two name parts is also divided into the same two name parts (tokens). In this

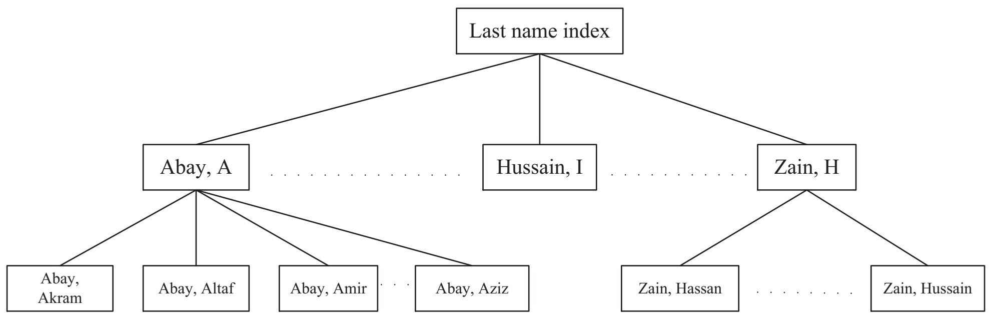

Figure 3. A blocking-based author names index structure.

indexing, Akram Abay in that block precedes Altaf Abay because the first name in Akram Abay alphabetically comes before that in Altaf Abay.

3.2. Authors profile builder

Disambiguated results are clusters of nodes (authors) in coauthor's graph. As mentioned in the previous section, the author node consists of node ID, node name, node publications. All author names are parsed into two components: first name and last name. We index all these names according to the proposed index in Section 3.1, using the last names of the authors.

When there is a huge number of existing authors, there is a very high probability that the new author belongs to some of the existing authors. The same intuition is used for other features of the citations like coauthors, titles, self-citations, and references. For example, an author usually publishes one's research with a specific set of coauthors, in certain venues, on certain topics, and self-cites very often [34]. According to a recent study conducted by King et al., 9.4% of all citations are self-citations [35]. Snyder and Bonzi estimated the pattern of self-citations in different disciplines and found that self-citation in physical sciences (15%) is larger than in social sciences (6%) and humanities (3%) [36]. In the light of these facts, we assume that self-citation may be a strong feature for author disambiguation.

Following these intuitions, we build a set of candidate author profile models, which we call author profile models. Recall from the preceding paragraphs that these author profile models have author names, set of coauthors, titles feature vectors, set of venues, and set of references. For example, we take an author Akram Abay, which is already disambiguated (batch AND) with the help of algorithms presented in [10]. Our batch AND algorithms keep a record of publications that belong to author Akram Abay. With these pieces of information, we can easily build the profiles of candidate authors that are equivalent (or similar) to the newly inserted authors. Figure 3 shows the structure of these author profiles. These profiles act as an author's signature and help to disambiguate authors.

3.3. Data extractor

When a number of new citations are inserted in BDs, citation records that are composed of a set of attributes such as authors, coauthors, titles, venues, and references of the papers are tokenized. These newly inserted records need disambiguation and assignment to their respective clusters. CAND takes these records one by one and retrieves a set of equivalent (similar) author profiles from the already disambiguated BDs. CAND constructs keyword feature vectors from the citation title words for comparison among author's research interests. In this research, it is assumed that such keyword feature vectors represent the author's research interests, at least

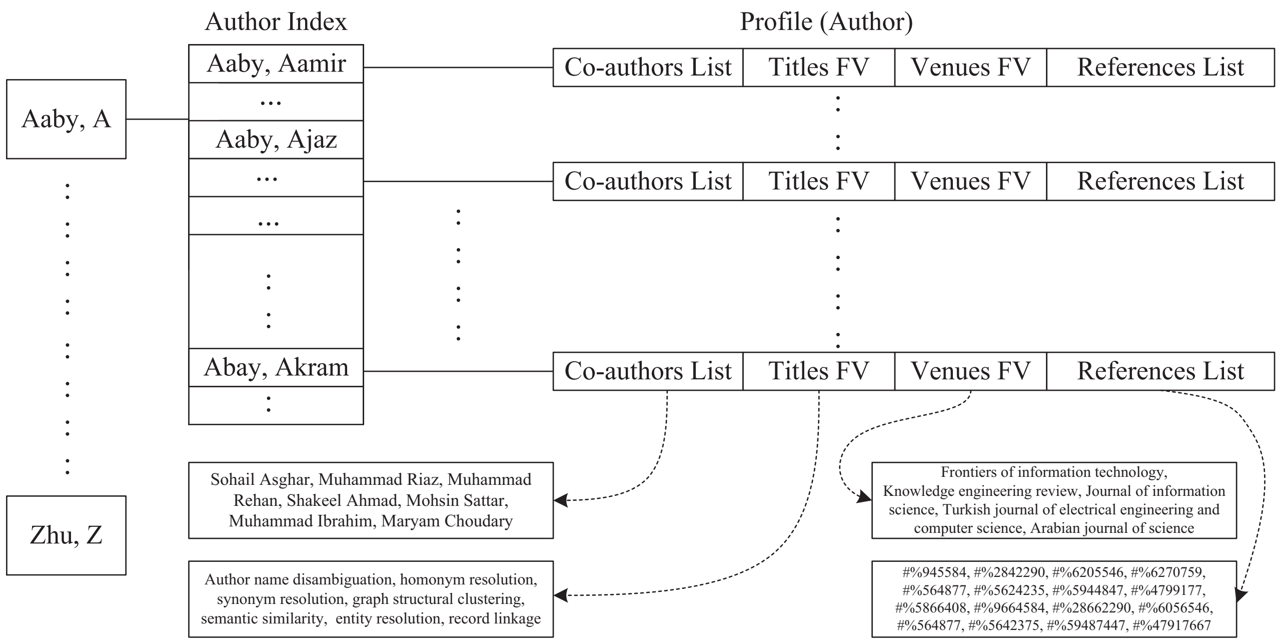
Author Index
Profile (Author)
Aaby, A
Zhu, Z
Aaby, Aamir
Aaby, Ajaz
Abay, Akram
Co-authors List
Titles FV
Venues FV
References List
Sohail Asghar, Muhammad Riaz, Muhammad Rehan, Shakeel Ahmad, Mohsin Sattar, Muhammad Ibrahim, Maryam Choudary
Author name disambiguation, homonym resolution, synonym resolution, graph structural clustering, semantic similarity, entity resolution, record linkage
Frontiers of information technology, Knowledge engineering review, Journal of information science, Turkish journal of electrical engineering and computer science, Arabian journal of science
#%945584, #%2842290, #%6205546, #%6270759, #%564877, #%5624235, #%5944847, #%4799177, #%5866408, #%9664584, #%28662290, #%6056546, #%564877, #%5642375, #%59487447, #%47917667

Figure 4. Author profiles from disambiguated BDs.

partially. However, such kind of title feature vector construction demands preprocessing the title words. In the titles, some words that are called stop words, for example, ‘a’, ‘an’,‘the’, ‘for’, ‘and’,‘of’,‘that’, are removed. These words usually do not carry any semantic information and are frequently used in titles. In addition, the remaining title keywords are stemmed. For example, the word “deputation” is transformed into the stem “deput” by a stemmer. An advantage of the stemming is that all morphological variants are transformed to the base word. Hence, it considerably reduces the dimensions of the feature vectors. CAND uses standard English Stop Word list, and Porter stemmer [37] for stemming.

3.4. Comparator

Given a set of candidate clusters and a new record, there are three possibilities that the newly inserted record either belongs to a candidate cluster or to more than one candidate clusters, or does not belong to any of the candidate clusters. Figure 5 illustrates an example of possible ways of assignments of new reference record to existing record of clusters. In the first case, the new record is merged with the equivalent (or similar) cluster and author profile is updated accordingly while in the second case, all the equivalent (or similar) clusters are merged into one cluster along with the new record, and author profiles are updated. When it is established with the help of the CAND algorithm that the new record does not belong to any existing cluster, then a new cluster is generated, and its author profile is built and saved for further disambiguation.

The pseudocode of the incremental author name ambiguity resolver algorithm is given in Algorithm 1. The algorithm starts with the preprocessing of the newly inserted citations as described in Section 3.3 of this paper (line 1). CAND retrieves the equivalent (or similar) author clusters to the newly inserted author in the cleaned BDs clusters and builds their profiles. These profiles are built and indexed for each cluster according to the procedures given in Sections 3.2 and 3.1, respectively. If there is no equivalent (or similar) cluster in the cleaned BDs, then a new cluster of the newly inserted reference is created and inserted in the cleaned BDs

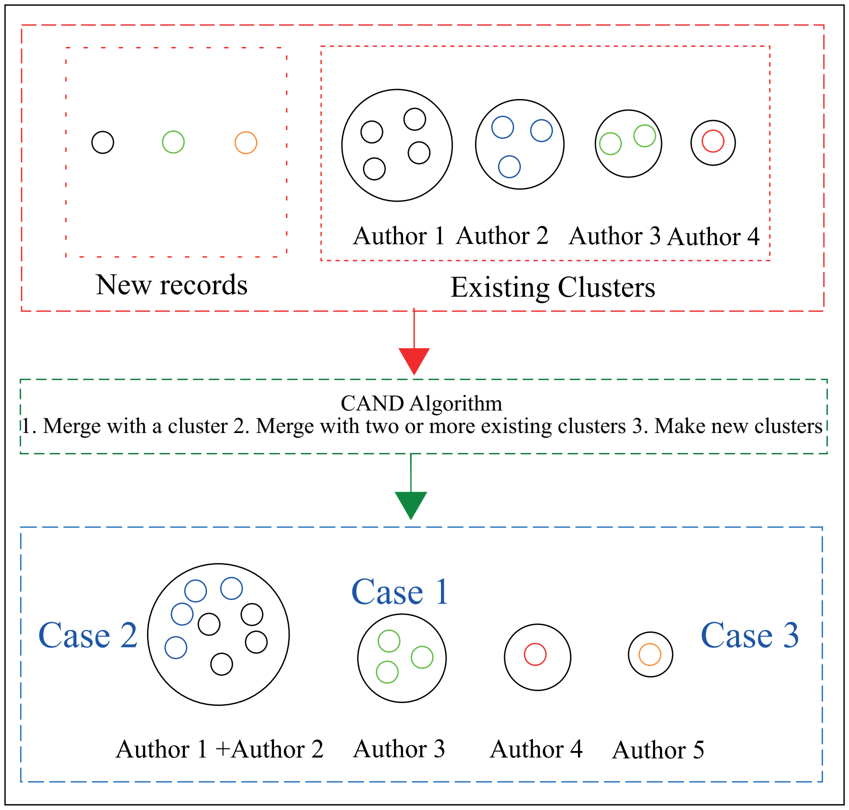

Figure 5. An example of new reference assignment to existing clusters.

(lines 2–11). Upon finding equivalent (or similar) clusters with respect to the newly inserted reference record using Jaro similarity as given in Equation 1 [38], CAND tests two conditions between all the clusters and the newly inserted records.

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

where c is the number of common characters in two name strings. A character that is considered a common character at position i in the string n_1 has to be within the H window of the equivalent j^{th} 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.

There are more than one similar coauthors present between the equivalent (or similar) cluster A_{cj} and newly inserted paper A_i using Equation 2:

$$Co_{auth}\ (A_i, A_{cj}) = \frac{|A_i \cap A_{cj}|}{min(|A_i|, |A_{cj}|)}. \tag{2}$$

The $\cap$ along with the enclosing $||$-operator finds the number of common coauthors. The second condition is that we find the self-citations using Equation 3, when the newly inserted paper cites some paper from the existing clusters.

Algorithm 1: CAND:Proposed Incremental AND Algorithm

```
Data: $Existingset\ of\ clusters\ C, A\ set\ of\ records\ R$
Result: $Set\ of\ disambiguated\ clusters\ DC$
1  $R' \longleftarrow\ Preprocess\ Records(R)$
2  $DC \longleftarrow\ 0$
3  for $refrence\ r \in R'$ do
4      $C' \longleftarrow Get\ Similar\ Clusters(C\ , r)$
5      $C'' \longleftarrow Build\ Author\ Profiles(C'\ , r)$
6      $P \longleftarrow Load\ Profiles(C''\ , r)$
7      if $P == null$ then
8          $c \longleftarrow create\ new\ cluster()$
9          $DC \longleftarrow C \cup c$
10         $Update(DC)$
11     end
12     for $cluster\ p \in P$ do
13         if $p.coauthors \cap r.coauthors \geqslant 1\ and\ Sim(P_{FV}, r_{FV}) > 0.1$
14         $or\ \ r\ cites\ p.publications$
15          then
16             $p = p \cup r$
17             $DC \longleftarrow C \cup p$
18             $Update(DC)$
19         end
20         else
21             $c \longleftarrow Create\ new\ Cluster(r)$
22             $DC \longleftarrow DC \cup c$
23             $Update(DC)$
24         end
25     end
26 end
27 $returnUpdatedClusters\ DC$
```

$$Self_{cit} = \sum_{j=1}^{n} \left(P_i \cap R_{cj}\right). \tag{3}$$

Feature vectors of titles are generated for all the papers in the data set. The similarity between the newly inserted paper title feature vector and cluster feature vector is found using Equation 4. The similarity index,

$$Similarity\ (A_{fv}, B_{fv}) = \frac{A_{fv} \cap B_{fv}}{max\,|(A_{fv}, B_{fv})|} \tag{4}$$

between these titles feature vectors is used for comparison between cluster and newly inserted paper. If any condition satisfies then CAND merges that newly inserted paper to the existing cluster and updates the cluster (lines 12–19). When these conditions are not satisfied, then a new cluster for the newly inserted paper is created and set of clusters is updated accordingly. At the end, a set of updated clusters is returned (lines 20–27).

4. Experiments and results

In this section, the performance of CAND is compared with two incremental AND techniques using Arnetminer and BDBComp in the form of clustering evaluation metrics. We used these data sets because they have citation information along with other attributes used in CAND and are open source data sets.

4.1. Data sets and evaluation measures

Tang et al. created Arnetminer data set from DBLP, ACM, and Microsoft academic graph, and other sources. Arnetminer has 629,814 papers and more than 632,752 citation relationships. It is organized into 600,000 blocks, one for each paper. Further details of Arnetminer can be viewed online at https://aminer.org/data. CAND uses only coauthors, titles, and citations for the complete solution of the incremental author name ambiguity problem.

BDBComp is a relatively small data set of 363 records belonging to 184 distinct authors, but it is very difficult to disambiguate as majority of the authors have only one or two citation records. BDBComp statistics are given in Table 2 and this collection also has been frequently used in similar AND studies [7, 8, 25].

Table 2. Details of BDBComp data set.

S No.	Name	No. of citation records	No. of ambiguous 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

We analyze the citations per author, which is also called the diversity of the data set. It is obtained by clustering the names of the authors in citations' data set. In Figure 6, the left figure shows the distribution of BDBComp, where 74.3% of the authors have published one paper. In contrast to this, in Arnetminer (right figure), 48.2% of the authors published only one paper.

Three clustering metrics, namely author cluster purity (ACP), Author average purity (AAP), and K-metric (K) as given in Equation 5, are used for evaluation of CAND [1, 9, 10].

$$ACP = \frac{1}{N}\sum_{r=1}^{R}\sum_{s=1}^{S}\frac{n_{rs}^2}{n_r}, AAP = \frac{1}{N}\sum_{s=1}^{S}\sum_{r=1}^{R}\frac{n_{rs}^2}{n_s}, K = \sqrt{ACP * AAP}, \tag{5}$$

where N is total number of references in ambiguous group, R is empirical clusters, S is ground truth clusters, n_{rs} is total references which are present in both empirical and in ground truth clusters, and n_r is total references present in empirical clusters.

4.2. Baseline methods

We found three incremental AND methods in the literature: INDi [7], reducing fragmentation in incremental author name disambiguation (INDi+) [8], and INC [9]. We choose these methods as baselines to compare with CAND because these are the state-of-the-art incremental AND methods and are closely related to our method.

Decarvalho et al., in [7], proposed INDi as a solution for the existing cleaned BDs. INDi utilized similarity between bibliographic records and groups of the records of authors with similar citation records in the BD or to new authors when the similarity evidence is not strong enough. INDi used coauthors, titles, and venues heuristics for checking whether references of new citation records belong to preexisting authors, or to new ones (i.e. authors without citation records in the BD). However, INDi suffered from two problems: first, it created fragmented clusters of authors (i.e. records of the equivalent author is split into multiple groups), and second, it could not handle the transitivity problem. Esperidiano et al., in [8], dropped the assumption that the existing BDs are cleaned, and they tried to merge the fragmented clusters that were produced by INDi. They also proposed different selection criteria to improve the cluster purity. Their strategy, which compared the newly inserted record with only the leading cluster record that is the closest to the centroid of each cluster, is considered the best overall performance in their experiments. We call this INDi+ and use this for comparison because it is an improved version of INDi. We set its parameters as $\alpha_{title} = 0.01$, $\alpha_{venue} = 0.1$ and $\delta = 0.6$.

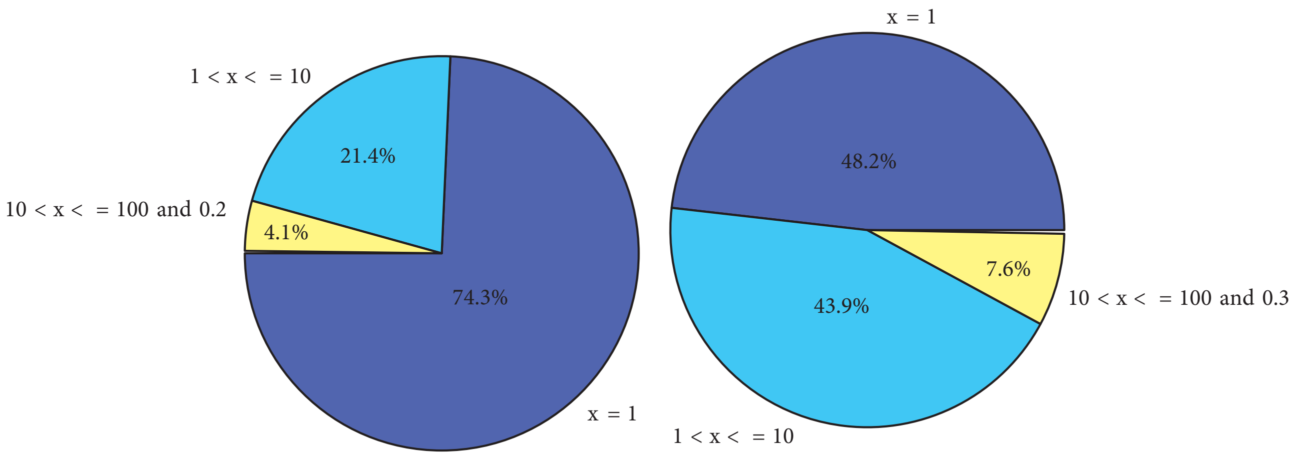

Figure 6. Author's citation distribution of BDBComp (left) and Arnetminer (right).

Recently, Santana et al., in [9], proposed INC that used fragment comparison method for retrieving the relevant author blocks to the new record, and then used author names, coauthor names, titles, and venue for similarity calculation between new records and the retrieved block of records. We use its best parameter values and set them as $W_a = 3, W_c = 3, W_t = 1, W_v = 1$ and $\gamma = 2$.

4.3. Experimental results and discussion

We used an incremental load from each year starting from 1987 to 2007. In 1987, whole BD was disambiguated using the batch AND algorithm proposed by us [18], whereas, for subsequent loads after 1987, CAND is used for each new year. Figure 7 shows CAND's performance on Arnetminer.

As seen in this figure, fragmentation is extremely low from 1987 to 1990 and from 2003 to 2007, but there is a slight decrease in between these two periods. Purity remains very high throughout 1987 to 2007, as it is one of the desirable characteristics of AND methods because in the later stages it is difficult or even impossible to split the merging error. At the end of the twenty-one-year period, the values of K, ACP, and AAP for CAND are 0.89, 0.87, and 0.92, respectively. On the other hand, as seen in Figure 8, purity is declining throughout the period from 1987 to 2007.

Fragmentation value is relatively stable and almost equal to the purity of CAND on Arnetminer. K, ACP, and AAP are 0.89, 0.78, and 0.83, respectively, for CAND using BDBComp at the end of the experiment. Another important aspect of the results is that CAND performance on Arnetminer is relatively higher, and in a narrower band than on BDBComp. We suspect that this is due to the fact that on average 56% of publications belong to new authors in BDBComp, whereas in Arnetminer only 34% belong to new authors. It is highly probable that the new inserted reference may be merged with some of the existing clusters.

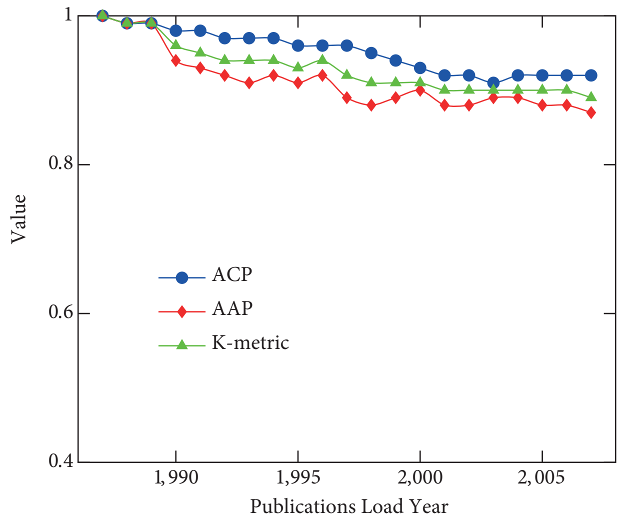

Figure 7. ACP, AAP, and K-metric performance evaluation of CAND on Arnetminer.

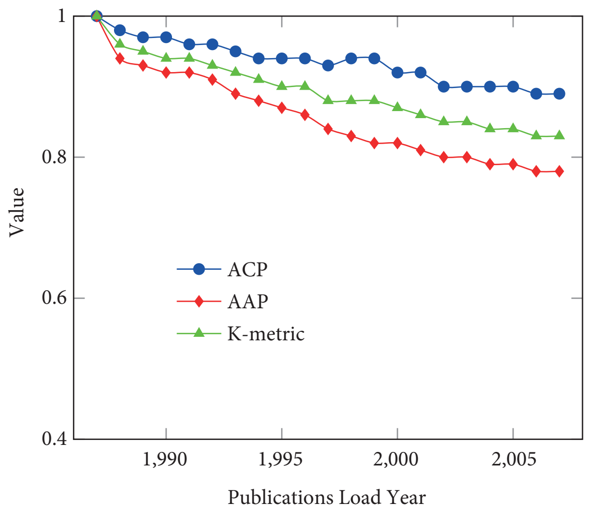

Figure 8. ACP, AAP, and K-metric performance evaluation of CAND on BDBComp.

Table 3 lists results using CAND, INDi+, and INC on Arnetminer for the period from 1987 to 2007. CAND achieves overall better performance than competing techniques during the whole period from 1987 to 2007. At the end of the twenty-one-year period, CAND achieves 14% and 11% higher value than INDi+ and INC in K metric using Arnetminer, respectively.

Similarly, Table 4 shows ACP, AAP, and K-metric of CAND, INDi+ and INC on BDBComp. CAND compares the new records with only the equivalent (or similar) cluster not with all the records in that cluster, so it improves the running time compared to INDi+ and INC, which compare the newly inserted reference to all the existing records. In general, ACP of CAND is higher than AAP because CAND produces pure clusters. CAND achieves overall better performance compared to INDi+ and INC during the whole period. At the end of the twenty-one-year period, CAND achieves 11% and 4% higher than INDi+ and INC in K-metric on BDBComp, respectively.

As seen, overall results of the CAND using Arnetminer are higher than the results of BDBComp. CAND performance using BDBComp is 6% less than Arnetminer. There are two main reasons behind this: first, in BDBComp on average 56% of the publications belong to new authors and the majority of authors has published only one article.

Table 3. Performance Evaluation of CAND, INDi+, and INC for each year starting from 1987 to 2007 on Arnetminer Collection

Arnetminer									
CAND				INDi+			INC		
Year	ACP	AAP	K	ACP	AAP	K	ACP	AAP	K
1987	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
1988	0.99	0.99	0.99	0.99	0.98	0.98	0.97	0.96	0.96
1989	0.99	0.99	0.99	0.98	0.96	0.97	0.96	0.95	0.95
1990	0.98	0.94	0.96	0.97	0.93	0.95	0.95	0.94	0.94
1991	0.98	0.93	0.95	0.97	0.92	0.94	0.95	0.95	0.95
1992	0.97	0.92	0.94	0.96	0.91	0.93	0.94	0.94	0.94
1993	0.97	0.91	0.94	0.95	0.90	0.92	0.92	0.93	0.92
1994	0.97	0.92	0.94	0.94	0.89	0.91	0.91	0.93	0.92
1995	0.96	0.91	0.93	0.93	0.87	0.90	0.91	0.92	0.91
1996	0.96	0.92	0.94	0.94	0.86	0.90	0.89	0.91	0.90
1997	0.96	0.89	0.92	0.92	0.85	0.88	0.87	0.90	0.88
1998	0.95	0.88	0.91	0.92	0.84	0.88	0.86	0.89	0.87
1999	0.94	0.89	0.91	0.91	0.83	0.87	0.85	0.89	0.87
2000	0.93	0.90	0.91	0.91	0.82	0.86	0.84	0.90	0.87
2001	0.92	0.88	0.90	0.91	0.81	0.86	0.82	0.91	0.86
2002	0.92	0.88	0.90	0.91	0.79	0.85	0.81	0.89	0.85
2003	0.91	0.89	0.90	0.91	0.78	0.84	0.80	0.87	0.83
2004	0.92	0.89	0.90	0.90	0.76	0.83	0.80	0.85	0.82
2005	0.92	0.88	0.90	0.90	0.74	0.82	0.79	0.83	0.81
2006	0.92	0.88	0.90	0.89	0.73	0.81	0.79	0.83	0.81
2007	0.92	0.87	0.89	0.88	0.70	0.78	0.79	0.82	0.80

The reason why CAND has better results than INDi+ and INC is that it exploits strong features; coauthors, content similarity, and self-citations. As INDi+ uses coauthor, titles, and venue similarity, some wrong merges and split of the publications on the basis of these similarities may occur. Although INC tries to improve the accuracy by comparing the new records with the similar existing records, they used the same coauthors, titles, and venues similarities, which did not improve the performance of the system because of the absence of discriminating training data.

CAND consists of four main stages in which building author profiles is a rather time-consuming step as compared to the other steps. However, it is worth noting that this is a one time process. Once the profiles of the authors are built, they are saved for subsequent retrievals and comparisons. CAND compares newly inserted records to similar clusters in contrast to all records, which is more efficient than competing methods. Table 5 shows the running times of the three incremental disambiguation methods using Arnetminer and BDBComp. All experiments were performed on a personal computer with Intel(R) Core(TM)i7-5200U CPU @ 2.20 GHz 2.20 GHz and 8 GB memory and all methods were implemented using Python 3.6.4. All times are an average run time of the twenty-one runs given in seconds.

Table 4. Performance evaluation of CAND, INDi+, and INC for each year starting from 1987 to 2007 on BDBComp

BDBComp									
CAND				INDi+			INC		
Year	ACP	AAP	K	ACP	AAP	K	ACP	AAP	K
1987	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
1988	0.98	0.94	0.96	0.96	0.98	0.97	0.97	0.96	0.96
1989	0.97	0.93	0.95	0.96	0.96	0.96	0.96	0.94	0.95
1990	0.97	0.92	0.94	0.95	0.93	0.94	0.98	0.92	0.95
1991	0.96	0.92	0.94	0.94	0.92	0.93	0.97	0.90	0.93
1992	0.96	0.91	0.93	0.93	0.91	0.92	0.95	0.90	0.92
1993	0.95	0.89	0.92	0.92	0.90	0.91	0.95	0.90	0.92
1994	0.94	0.88	0.91	0.90	0.89	0.89	0.96	0.89	0.92
1995	0.94	0.87	0.90	0.91	0.87	0.89	0.94	0.88	0.91
1996	0.94	0.86	0.90	0.87	0.86	0.86	0.95	0.88	0.91
1997	0.93	0.84	0.88	0.88	0.85	0.86	0.93	0.87	0.90
1998	0.94	0.83	0.88	0.87	0.84	0.85	0.97	0.86	0.91
1999	0.94	0.82	0.88	0.86	0.83	0.84	0.96	0.85	0.90
2000	0.92	0.82	0.87	0.85	0.82	0.83	0.94	0.85	0.89
2001	0.92	0.81	0.86	0.82	0.81	0.81	0.92	0.84	0.88
2002	0.90	0.80	0.85	0.80	0.80	0.80	0.91	0.82	0.86
2003	0.90	0.80	0.85	0.79	0.79	0.79	0.90	0.80	0.85
2004	0.90	0.79	0.84	0.77	0.77	0.77	0.90	0.79	0.84
2005	0.90	0.79	0.84	0.76	0.77	0.76	0.89	0.78	0.83
2006	0.89	0.78	0.83	0.75	0.77	0.76	0.88	0.76	0.82
2007	0.89	0.78	0.83	0.74	0.77	0.75	0.87	0.74	0.80

Table 5. Running times comparison among CAND, INDi+, and INC on Arnetminer and BDBComp.

Method			
Data Set	CAND	INDi+	INC
Arnetminer	434.098 ± 20.054	684.847 ± 34.524	875.421 ± 43.867
BDBComp	0.296 ± 0.087	0.387 ± 0.094	0.514 ± 0.108

As seen in Table 5, INC is the slowest among all the methods on both data sets. INC calculates prior probability distributions of all the attributes which is a time consuming step. CAND is faster than the other techniques due to its indexing structure and comparing newly inserted records with the same clusters (not all records). CAND is about 1.6 to 2.1 times faster than INDi+ and INC, respectively. CAND fails on very ambiguous authors cases as an example is given in Table 6. In different citations, if two ambiguous authors share ambiguous coauthors, then these ambiguous authors are called very ambiguous authors as shown in Table 6, the first two publications belong to two different "Chun Chen" and "Bing Liu" and last two ambiguous authors "Chuen-Liang Chen" and "C. Chen" share ambiguous coauthors "Biing-Feng Wang" and "B. Wang".

Table 6. A very ambiguous author case.

Publications		
Citation Id	Authors	Title of Publication
C1	**Chun Chen**, Guang Qiu, Jiajun Bu, **Bing Liu**	Expanding domain sentiment lexicon through double propagation
C2	Robert F. Lucas, Mary W. Hall, Jacqueline Chame, **Chun Chen**, Nastaran Baradaran, Yoon-Ju Lee, **Bing Liu**, Pedro C. Diniz	ECO: An empirical-based compilation and optimization system
C3	**Chuen-Liang Chen**, **Biing-FengWang**, Gen-Huey Chen	A Simple Approach to Implementing Multiplication with Small Tables
C4	**C. Chen**, Yang Xiao, **B.Wang**	Bandwidth Degradation QoS for Adaptive Multimedia inWireless/Mobile Networks

5. Conclusions and future work

Self-citations and author profiles are little or never used in the domain of author name disambiguation. In this paper, we propose CAND, which solves the incremental author name ambiguity problem in ever-growing BDs using coauthors, titles, author profile models, and self-citations. CAND is an unsupervised method that neither require costly training data nor a priori hidden information such as the number of ambiguous authors. CAND performance is tested on two real-world benchmark data sets of Arnetminer and BDBComp. It shows overall better results than incremental baseline techniques. CAND delivers an effective (in term of clustering metrics) solution to the incremental author name ambiguity problem by using coauthors, content similarity, and citation networks. In the future, we intend to use some strategies to automatically find thresholds for different parameters of CAND.

Acknowledgment

We are thankful to Higher Education Commission (HEC) of Pakistan for partial funding of this research under the Indigenous fellowship 2PS2-566.

References

[1] 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 2014; 100(1): 15-50. doi: 10.1007/s11192-014-1289-4

[2] Han H, Xu W, Zha H, Giles CL. A hierarchical naive Bayes mixture model for name disambiguation in author citations. In: Proceedings of the 2005 ACM symposium on Applied computing; Santa Fe, NM, USA; 2005. pp. 1065-1069.

[3] Han D, Liu S, Hu Y, Wang B, Sun Y. Elm-based name disambiguation in bibliography. World Wide Web 2015; 18 (2): 253-263. doi: 10.1007/s11280-013-0226-4

[4] Bollen J, Rodriguez MA, Van de Sompel H, Balakireva LL, Hagberg A. The largest scholarly semantic network ever. In: Proceedings of the 16th international conference on World Wide Web 2007; Banff, Alberta, Canada. pp. 1247-1248.

[5] Jinha AE. Article 50 million: an estimate of the number of scholarly articles in existence. Learned Publishing 2010; 23(3): 258-263. doi: 10.1087/20100308

[6] Hussain I, Asghar S. A survey of author name disambiguation techniques: 2010–2016. Knowledge Engineering Review 2017; 32. doi: 10.1017/S0269888917000182

[7] De Carvalho A P, Ferreira A A, Laender A H, Gon calves M A. Incremental unsupervised name disambiguation in cleaned digital libraries. Journal of Information and Data Management 2011; 2(3): 289.

[8] Esperidiao LVB, Ferreira AA, Laender AH, Goncalves MA, Gomes DM et al. Reducing fragmentation in incremental author name disambiguation. Journal of Information and Data Management 2014; 5 (3): 293.

[9] Santana AF, Gonçalves MA, Laender AH, Ferreira AA. Incremental author name disambiguation by exploiting domain-specific heuristics. Journal of Association of Information Science and Technology 2017; 68(4): 931-945. doi: 10.1002/asi.23726

[10] Hussain I, Asghar S. DISC: Disambiguating homonyms using graph structural clustering. Journal of Information Science 2018; Journal of Information Science, 44(6), 830-847. doi: 10.1177/0165551518761011

[11] Wang J, Berzins K, Hicks D, Melkers J, Xiao F et al. A boosted-trees method for name disambiguation. Scientometrics 2012; 93 (2): 391-411. doi: 10.1007/s11192-012-0681-1

[12] Tran HN, Huynh T, Do T. Author name disambiguation by using deep neural network. In: Asian Conference on Intelligent Information and Database Systems 2014; Cham; 2014. pp. 123-132.

[13] Shoaib M, Daud A, Khiyal M. Improving Similarity Measures for Publications with Special Focus on Author Name Disambiguation. Arabian Journal for Science and Engineering 2015; 40(6) : 1591-1605. doi: 10.1007/s13369-015-1636-7

[14] Tang J, Fong AC, Wang B, Zhang J. A unified probabilistic framework for name disambiguation in digital library. IEEE Transactions on Knowledge and Data Engineering 2012; 24 (6): 975-987. doi: 10.1109/TKDE.2011.13

[15] Cota RG, Ferreira AA, Nascimento C, Gonçalves MA, Laender AH. An unsupervised heuristic-based hierarchical method for name disambiguation in bibliographic citations. Journal of the American Society for Information Science and Technology 2010; 61(9): 1853-1870. doi: 10.1002/asi.21363

[16] Wu H, Li B, Pei Y, He J. Unsupervised author disambiguation using dempster-shafer theory. Scientometrics 2014; 101 (3): 1955-1972. doi: 10.1007/s11192-014-1283-x

[17] Hussain I, Asghar S. Resolving namesakes using the author's social network. Turkish Journal of Electrical Engineering & Computer Science 2018; 26(1): 554-569. doi:10.3906/elk-1702-293

[18] Hussain I, Asghar S. Author name disambiguation by exploiting graph structural clustering and hybrid similarity. Arabian Journal for Science and Engineering 2018; 1-17. doi: 10.1007/s13369-018-3099-0

[19] Onodera N, Iwasawa M, Midorikawa N, Yoshikane F, Amano K et al. 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 2011; 62 (4): 677-690. doi: 10.1002/asi.21491

[20] Imran M, Gillani S, Marchese M. A real-time heuristic-based unsupervised method for name disambiguation in digital libraries. D-Lib Magazine 2013; 19 (9): 1. doi: 10.1045/september2013-imran

[21] Zhu J, Yang Y, Xie Q, Wang L, Hassan SU. Robust hybrid name disambiguation framework for large databases. Scientometrics 2014; 98 (3): 2255-2274. doi: 10.1007/s11192-013-1151-0

[22] Louppe G, Al-Natsheh HT, Susik M, Maguire EJ. Ethnicity sensitive author disambiguation using semi-supervised learning. In: International Conference on Knowledge Engineering and the Semantic Web 2016; Springer, Cham, 2016. pp. 272-287.

[23] Fan X, Wang J, Pu X, Zhou L, Lv B. On graph-based name disambiguation. Journal of Data and Information Quality 2011; 2 (2): 10. doi: 10.1145/1891879.1891883

[24] Wang X, Tang J, Cheng H, Philip SY. Adana: Active name disambiguation. In: 2011 IEEE 11th International Conference on Data Mining 2011 (ICDM); IEEE Vancouver, BC, Canada. pp. 794-803.

[25] Levin FH, Heuser CA. Evaluating the use of social networks in author name disambiguation in digital libraries. Journal of Information and Data Management 2010; 1(2): 183.

[26] Pereira DA, Ribeiro-Neto B, Ziviani N, Laender AH, Gonçalves MA et al. Using web information for author name disambiguation. In: Proceedings of the 9th ACM/IEEE-CS joint conference on Digital libraries 2009 (JCDL); Austin, TX, USA. pp. 49-58.

[27] Veloso A, Ferreira AA, Gonçalves MA, Laender AH, Meira Jr W. Cost-effective on-demand associative author name disambiguation. Information Processing & Management 2012; 48(4):680-697. doi: 10.1016/j.ipm.2011.08.005

[28] Wu J, Ding X H. Author name disambiguation in scientific collaboration and mobility cases. Scientometrics 2013; 96(3): 683-697. doi: 10.1007/s11192-013-0978-8

[29] Müller M C, Reitz F, Roy N. Data sets for author name disambiguation: an empirical analysis and a new resource. Scientometrics 2017;111(3):1467-1500. doi: 10.1007/s11192-017-2363-5

[30] Kim J. Evaluating author name disambiguation for digital libraries: a case of DBLP. Scientometrics. 2018; 116(3): 1867-1886. doi: 10.1007/s11192-018-2824-5

[31] Abdulhayoglu MA, Thijs B. Use of ResearchGate and Google CSE for author name disambiguation. Scientometrics 2017; 111(3): 1965-1985. doi: 10.1007/s11192-017-2341-y

[32] Zhao Z, Rollins J, Bai L, Rosen G. Incremental author name disambiguation for Scientific Citation Data. In: IEEE International Conference on Data Science and Advanced Analytics (DSAA) 2017; Tokyo, Japan 2017. pp. 175-183.

[33] Kim J, Kim J, Owen-Smith J. Generating automatically labeled data for author name disambiguation: an iterative clustering method. Scientometrics 2018; 1-28. doi: 10.1007/s11192-018-2968-3

[34] Hellsten I, Lambiotte R, Scharnhorst A, Ausloos M. Self-citations, co-authorships and keywords: a new approach to scientists' field mobility?. Scientometrics 2007; 72(3): 469-486. doi: 10.1007/s11192-007-1680-5

[35] King MM, Bergstrom CT, Correll SJ, Jacquet J, West JD. Men set their own cites high: Gender and self-citation across fields and over time. Socius 2017; 3: 2378023117738903. doi: 10.1177/2378023117738903

[36] Snyder H, Bonzi S. Patterns of self-citation across disciplines (1980-1989). Journal of Information Science 1998; 24(6): 431-435. doi: 10.1177/016555159802400606

[37] Porter MF. An algorithm for suffix stripping. PROGRAM 1980; 14(3): 130-137. doi: 10.1108/eb046814

[38] Jaro MA. Advances in record-linkage methodology as applied to matching the 1985 census of Tampa, Florida. Journal of the American Statistical Association 1989; 84(406): 414-420. doi: 10.1080/01621459.1989.10478785