Inf Retrieval J (2015) 18:379–412
DOI 10.1007/s10791-015-9261-3

Dynamic author name disambiguation for growing digital libraries

Yanan Qian[1] · Qinghua Zheng[1] · Tetsuya Sakai[2] · Junting Ye[1] · Jun Liu[1]

Received: 4 November 2014 / Accepted: 7 July 2015 / Published online: 21 July 2015

Abstract When a digital library user searches for publications by an author name, she often sees a mixture of publications by different authors who have the same name. With the growth of digital libraries and involvement of more authors, this author ambiguity problem is becoming critical. Author disambiguation (AD) often tries to solve this problem by leveraging metadata such as coauthors, research topics, publication venues and citation information, since more personal information such as the contact details is often restricted or missing. In this paper, we study the problem of how to efficiently disambiguate author names given an incessant stream of published papers. To this end, we propose a "BatchAD+IncAD" framework for dynamic author disambiguation. First, we perform batch author disambiguation (BatchAD) to disambiguate all author names at a given time by grouping all records (each record refers to a paper with one of its author names) into disjoint clusters. This establishes a one-to-one mapping between the clusters and real-world authors. Then, for newly added papers, we periodically perform incremental author disambiguation (IncAD), which determines whether each new record can be assigned to an existing cluster, or to a new cluster not yet included in the previous data. Based on the new data, IncAD also tries to correct previous AD results. Our main contributions are: (1) We demonstrate with real data that a small number of new papers often have overlapping author names with a large portion of existing papers, so it is challenging for IncAD to effectively leverage previous AD results. (2) We propose a novel IncAD model which aggregates metadata from a cluster of records to estimate the author's profile such as her coauthor distributions and keyword distributions, in order to predict how likely it is that a new record is "produced" by the author. (3) Using two labeled datasets and one large-scale

The paper is original, its content has never been published elsewhere before.

✉ Yanan Qian
yaya213@gmail.com

1 Department of Computer Science and Technology, Xi'an Jiaotong University, Xi'an, China

2 Department of Computer Science and Engineering, Waseda University, Tokyo, Japan

raw dataset, we show that the proposed method is much more efficient than state-of-the-art methods while ensuring high accuracy.

Keywords Digital library · Author disambiguation · Data stream · Clustering · Multi-classification

1 Introduction

Information on the Web often lacks uniform resource identifiers and causes ambiguity problems. When a digital library (e.g. DBLP, Citeseer, Google Scholar, and Microsoft Academic Search) user searches for publications by an author name, she often sees a mixture of publications by different authors who have the same name. With the growth of digital libraries and involvement of more authors, this author ambiguity problem is becoming critical. It is estimated that 50 million academic papers have been published so far, and that new papers are emerging as fast as “one paper per minute” on average (Jinha 2010). Moreover, the growth rate of academic papers is increasing (Bollen et al. 2007).

Author disambiguation (AD) often tries to solve the above problem by leveraging metadata such as coauthors, research topics, publication venues, citation information and download links, since more personal information such as contact details is often restricted or missing in the available data. In a digital library, we do not know in advance which author names are ambiguous, so that AD algorithms need to process all author names in all papers, which can be very time-consuming. Although a lot of effort has been devoted to improve the efficiency of AD (Song et al. 2007; Byung-won and Lee (2007; Wang et al. 2008; Huang et al. 2006), most of them did not take into account the fact that publication data are an ever-growing, incessant stream rather than just a static block of data. In this “batch-only” setting, an AD algorithm will have to re-disambiguate the entire data from scratch when new papers arrive. As this is time-consuming, frequent updating of the AD results would not be practical.

In this paper, we propose a novel “BatchAD+IncAD” dynamic author disambiguation method for the AD problem: first, we perform batch author disambiguation (BatchAD) to disambiguate all author names at a given time; then, we periodically perform Incremental Author Disambiguation (IncAD) for the author names in newly arrived papers. In the BatchAD phase, we propose an unsupervised method since no prior knowledge is available. Specifically, BatchAD groups all records (each record refers to a paper with one of its author names) into disjoint clusters, in order to establish a one-to-one mapping between the clusters and real-world authors. In the IncAD phrase, we propose to utilize the previous AD results to guide the disambiguation of author names in new papers.

IncAD faces two major challenges: The first is how to leverage previous AD results. At a first glance, this problem setting is somewhat similar to the multi-classification model proposed in Han et al. (2004). The model regards each cluster as a class and tries to assign each record to one of the classes. However, the problem of IncAD is different from this in two aspects: first, a record may not belong to any existing author when its author name in fact represents a new author; second, the previous AD results may contain some noise, which may hamper accurate processing of the new data.

The second challenge of IncAD is efficiency. In digital libraries, there are usually a small number of highly ambiguous names such as “W. Wang” and “Lei Zhang” which occupy the majority of the computational cost in AD. In one update, even if only a few papers are added, these highly ambiguous author names are often involved, which correspond to a significant part of the existing data (as will be quantitatively analyzed in Sect. 2.2). This property might make IncAD a time-consuming procedure.

In light of the above challenges, we propose a novel IncAD approach by modeling each record cluster (representing an author) in previous AD results as an *author model*. In the "author model", we aggregate the metadata from individual records, in order to capture the information such as "which coauthors is the author likely to have?" and "what keywords is the author likely to use?". As a result, each author model is able to predict how likely it is that a new record is "produced" by the author, and decide whether to assign the new record to the record cluster. When disambiguating the author name of a new record, IncAD first loads all author models with this author name, and then predicts the assignment of the record. Three types of updates could possibly happen: (1) If the record cannot be assigned to any existing cluster, its author name probably represents a new author, so we create a new cluster for it; (2) If the record is mapped to exactly one existing cluster, we will execute the assignment, which indicates the record belongs to an identified author. (3) If the record could be mapped to two or more existing clusters, this suggests that these clusters (and their corresponding author models) represent the same author and therefore they need to be merged. Also, the record should be assigned to the merged cluster.

There are two major advantages of IncAD over existing methods. First, after processing an update (i.e. a set of new records), the author models can be easily updated based on new evidence observed in the new records, so that IncAD can get ready for future updates. On the contrary, existing methods often rely on fixed parameters or fixed heuristics along with the updating process. Second, the IncAD complexity is proportional to the number of existing clusters instead of the number of individual records. Since the number of clusters is usually much smaller than that of records, it is computationally more efficient than existing methods that compare new records with existing records directly.

Using two labeled data sets and one large-scale unlabeled data set, we have conducted extensive experiments to evaluate the efficiency and accuracy of our proposed method. Our main findings are: (1) In terms of efficiency, "BatchAD + IncAD" far outperforms the BatchAD-only approach, while BatchAD outperforms five state-of-the-art batch methods (it only costs 59.67 % time of the best baseline) and IncAD far outperforms three alternative incremental AD methods (it only costs 31.45 % time of the best baseline). (2) In terms of accuracy, "BatchAD + IncAD" is comparable to the BatchAD-only method (with only less than 3 % loss) after processing 10 years' worth of weekly paper updates. Meanwhile, BatchAD shows comparable performance with the best state-of-the-art method. IncAD far outperforms two baselines and is comparable to another baseline which is equal to BatchAD in terms of accuracy but is poor in terms of efficiency. (3) According our IncAD's results, 57.71 % of new records are assigned to existing clusters, 23.94 % are assigned to new clusters, and the other 18.35 % cause corrections to the previous AD results and are then assigned to the corrected clusters.

The major contributions of this paper are:

- First, we demonstrate with real data that a small number of new papers often have overlapping author names with a large portion of existing papers, so it is challenging for IncAD to effectively leverage previous AD results.
- Second, we propose a novel BatchAD+IncAD framework for dynamic author disambiguation. In BatchAD, we disambiguate all author names without any prior knowledge. In IncAD, we model each record cluster as a generative author model so

that it can effectively predict the assignments of new records. IncAD can also efficiently update the generative author models given the information with the newly assigned records, so that IncAD could always be ready for future updates.
- Third, through multiple experiments, we show that the proposed method is much more efficient than state-of-the-art methods while ensuring high accuracy.

The rest of the paper is organized as follows. Section 2 discusses the dynamic properties of author disambiguation and identifies requirements for AD algorithms. Section 3 discusses related work. Section 4 describes our BatchAD and IncAD algorithms, and Sect. 5 presents experimental results and discussions. Finally, Sect. 6 concludes the paper and proposes future work.

2 Motivation and analysis

In this section, we first present statistics on the growth of a digital library and those on useful metadata for AD. We then discuss challenges of ensuring AD accuracy and efficiency in the dynamic settings. Finally we conclude the section by identifying some requirements for our BatchAD and IncAD algorithms.

2.1 Growth of the digital library

To identify the requirements of practical AD algorithms, we observed the data growth of Microsoft Academic Search over a year (Nov. 2011–Jan. 2013), as shown in Fig. 1. In this period, the number of papers expanded from 37 to 49 million. On average, 119,301 new papers were added each week, which indicates that the data size was enlarging by 0.25 % per week; in the fastest growing week, 1,994,381 papers (4.07 %) were added. Since the scale of the new data is small compared to the existing data, it is not economical to run a batch AD algorithm periodically.

To perform AD accurately, we should leverage the metadata (including author name, coauthor names, paper title, etc.) of a record whenever available, since we may not be able to obtain the authors' personal information from the papers. To illustrate this point, Fig. 2 shows the coverage of different metadata for a sample of the Microsoft Academic Search data. The sample contains 3.5 million papers and 0.5 million distinct author names, which will be described as the "Unlabeled" dataset in Sect. 5.1.1. All analysis in the remainder of this section will be based on this dataset. Figure 2 shows that the metadata at any given

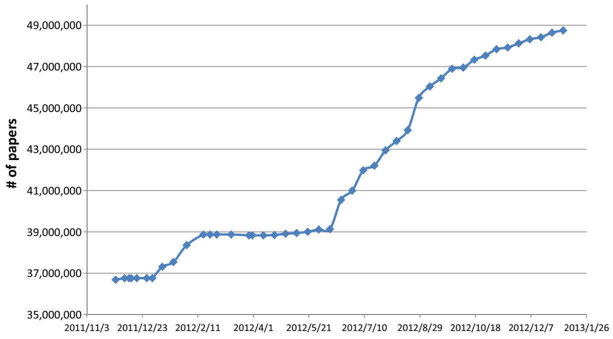

Fig. 1 The data growth of Microsoft Academic Search

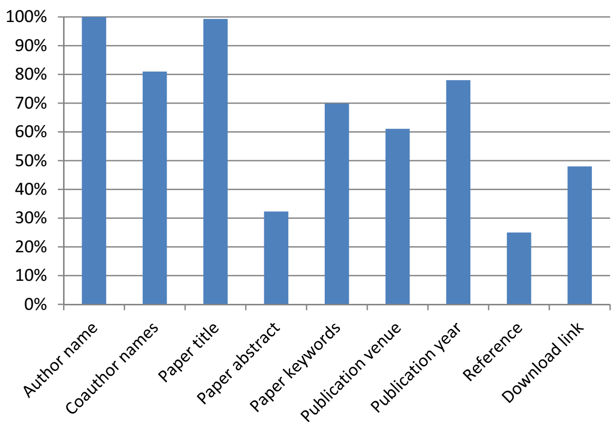

Fig. 2 The coverage of different metadata

time are incomplete, so it is probably wise for an AD algorithm to utilize multiple metadata that complement each other.

Besides, the metadata are also constantly evolving: the coauthor network grows every time an author writes a new paper with a new collaborator; the reference network grows every time there is a new paper citation; an author might upload her papers on her personal webpage which provides more download links for the papers, etc. Since the metadata are ever-growing, an incremental AD algorithm should be able to correct previous AD results based on new evidence (e.g. observation of new metadata).

2.2 Challenges in dynamic author disambiguation

Next, we will identify the challenges of author disambiguation in an ever-growing digital library. First, we will analyze the precision errors and recall errors which are two typical cases that can hurt the AD accuracy. Then, we will investigate the data scale and data distribution which determine the potential computational complexity (and efficiency) of AD methods. Below, we discuss each in turn.

Precision error refers to cases where the records belong to different authors are incorrectly put into the same cluster. This is particularly challenging, for example, if multiple authors have the same name, work at the same institution and/or share similar research interests. Consider the precision problem in an incremental AD setting: when a new author emerges with a small number of papers, and she happens to have the same name with some known authors who have published many papers, it might be challenging to identify the new author accurately.

Recall error refers to cases where records belonging to the same author are incorrectly split into different clusters. Table 1 shows an example where five records from the same author might be separated into two clusters as there appears to be two different research topics. Thus if a digital library user searches for publications by Zaiqing Nie, the system may present the first cluster only, resulting in low recall. Recall error can happen when: (1) an author works on multiple research topics as in this example; (2) an author changes her job, which often results in new research topics and new coauthors; and (3) some records have incomplete metadata, so that there is not enough evidence to merge them. Previous work (Byung-won et al. 2006; Yin et al. 2007; Amigó et al. 2009; Tang et al. 2012)

Table 1 The same "Zaiqing Nie" with two groups of papers

Papers
WebPage Understanding: Beyond Page-Level Search
Zaiqing Nie, Ji-Rong Wen, and Wei-Ying Ma. (SIGMOD 2008)
Web Object Retrieval
Zaiqing Nie, Yunxiao Ma, Shuming Shi, Ji-Rong Wen, Wei-Ying Ma. (WWW 2007)
2D Conditional Random Fields for Web Information Extraction
Jun Zhu, *Zaiqing Nie*, Ji-Rong Wen, Bo Zhang, Wei-Ying Ma. (ICML 2005)
Effectively Mining and Using Coverage and Overlap Statistics for Data Integration
Zaiqing Nie, Subbarao Kambhampati and Ullas Nambiar. (TKDE 2005)
A Frequency-based Approach for Mining Coverage Statistics in Data Integration
Zaiqing Nie, Subbarao Kambhampati (ICDE 2004)

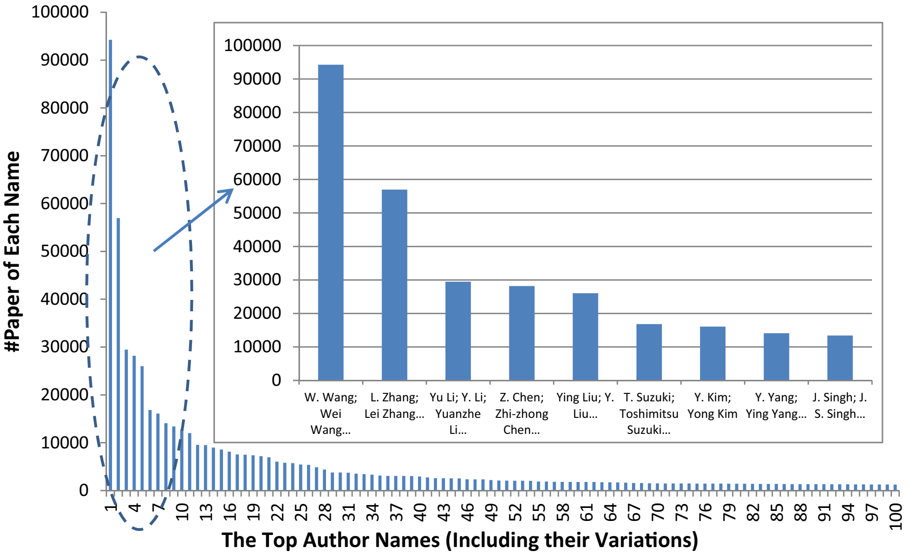

Fig. 3 The author name distribution in regards of paper count (on a Microsoft Academic Search data sample with 3.5 million papers)

shows that recall is generally much lower than precision for AD. Thus it is very important for a practical AD algorithm to achieve high recall.

Now we discuss the potential computational complexity of AD algorithms. A digital library often contains a large set of papers with various author names. In the batch setting, an AD algorithm does not know in advance which author names are ambiguous, so it needs to process all author names. In fact, a few common names often account for a large portion of the papers (as illustrated in Fig. 3), so disambiguating those common names becomes the performance bottleneck. A practical AD algorithm should handle such highly ambiguous names efficiently.

We further investigate the incremental setting: consider a digital library where all author names have been disambiguated, and suppose that one new paper is added to it. If we want

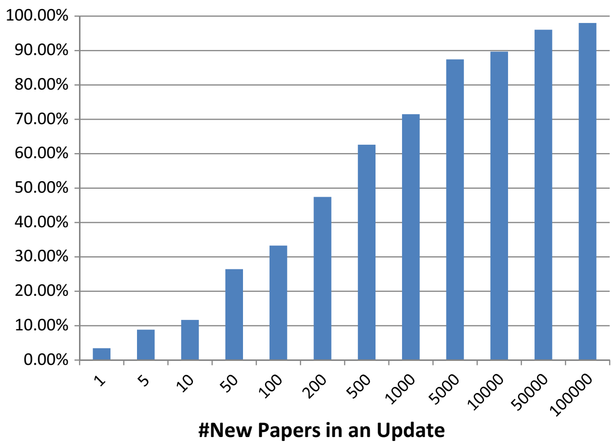

Fig. 4 Percentage of existing papers which has overlapping author names with the newly added 1, 5, 10... papers (on a Microsoft Academic Search data sample with 3.5 million papers)

to disambiguate the author names in this paper without batch-processing the entire data, we need to find whether there are identical author names (or similar author names such as an abbreviated name) that already appeared and were disambiguated in existing data. Since a small number of common author names cover a large fraction of papers in digital libraries, the chance that a new paper contains a common name that overlaps with some existing author names is high. As a result, even for an update consisting of a small number of papers, the author names in these new papers might overlap with a lot of papers in the past data.

We use the Microsoft Academic Search data sample to verify the assumption. We randomly chose one paper from the set and regarded it as new, and we found that on average about 100,000 papers from the remaining set have at least one author name that overlaps with this paper. As we further enlarge the number of new papers to 5, 10, 100 and more, the percentage of existing papers with overlapping author names grows quickly, as Fig. 4 shows. When regarding 1000 papers (0.028 % of all papers) as new, the percentage goes up to 85 %. This result indicates that even when incrementally disambiguating the author names in a small number of new papers (i.e. incrementally clustering a small number of records), a large portion of existing clusters need to be considered as candidates, thus the potential computational complexity of IncAD might also be high.

2.3 Design philosophy

Having studied the properties of digital library data and the challenges in AD, we decided to design a system that comprises a batch AD algorithm (BatchAD) and an incremental AD algorithm (IncAD). BatchAD is performed only once to process existing data, and IncAD is performed periodically to process new data. Since digital libraries are growing fast (as we have seen in Sect. 2.1), our incremental approach seems to be a natural and practical choice to process the incoming data in a timely manner. More specifically, we established the following design philosophy for BatchAD and IncAD.

- *Indexing* BatchAD should index AD results in such a way that IncAD can efficiently locate the known AD results of a given author name that appears in the new papers.
- *For accuracy* BatchAD should maintain high precision rather than high recall. This is because correcting precision errors in AD results by splitting record clusters is much

more difficult and costly to manage than correcting recall errors by merging them. On the other hand, IncAD should help BatchAD correct recall errors.

- *For efficiency* Since a new paper is likely to have overlapping author names with a large portion of existing papers, and those overlapping author names may refer to a large set of record clusters, it is essential to model each cluster in a way that it can predict whether a new record can be assigned to it efficiently. Also, IncAD should enable mini-batch updates (i.e., frequent updating with a small number of papers).

3 Related work

In the literature, author disambiguation has been formulated as multi-classification problems, clustering problems as well as others. Below, we briefly survey each in turn.

The multi-classification model regards each author as a class, and aims at classifying records into the classes. It requires prior knowledge of a record database for each author as training data. Under this framework, Han et al. (2004) proposed two supervised learning approaches including a Naive Bayes based classifier and a SVM based classifier; Nguyen and Cao (2008) proposed to assign extracted named entities to an established knowledge base such as Wikipedia.

In the absence of prior knowledge and training data about authors, unsupervised clustering approaches are necessary. Under this framework, Han et al. (2003) proposed to apply k-means clustering to AD by partitioning n records into k clusters in which each record belongs to the cluster with the nearest distance. They also proposed a k-way spectral clustering based AD (Han et al. 2005). However, such methods require an accurate estimate of the actual number of authors k. Tang et al. (2012) adopted the Bayesian Information Criterion for estimating k. Also, hierarchical agglomerative clustering is widely used for AD (Song et al. 2007; Yin et al. 2007; Pereira et al. 2009; Monz and Weerkamp 2009; Torvik and Smalheiser 2008), which works in a bottom-up fashion and merges the most similar record clusters in each iteration. For example, Yin et al. (2007) proposed DISTINCT, which leverages the average-linked agglomerative clustering algorithm. Gong and Oard (2009) proposed a method for selecting an appropriate threshold for the hierarchical clustering process in AD. Huang et al. (2006) applied a density based clustering algorithm DBSCAN for AD, which constructs clusters based on pairwise distances and guarantees the transitivity for core points. Zhang et al. (2010) employed affinity propagation clustering for AD. Balog et al. (2009) compared four clustering algorithms including single pass clustering, k-means clustering, agglomerative clustering and Probabilistic Latent Semantic Analysis (PLSA) for AD, and reported that agglomerative clustering performed best.

Moreover, various other models have also been proposed. Bhattacharya and Getoor (2007) proposed a collective entity disambiguation method which disambiguates less-ambiguous names first, and then use them as evidence to disambiguate their more-ambiguous coauthors; this process is iterated until all names are disambiguated. Tang et al. (2012) proposed to encode record features and multiple record relationships into a unified probabilistic framework. Moreover, several graph partition based methods (Spencer and Shadbolt 2006; Jiang et al. 2009; Fan et al. 2008) have been proposed.

A key issue in the above approaches is modeling the similarity between records. A typical feature set of “title, author, venue and publication year” was widely adopted (Han et al. 2004, 2003; Na et al. 2009). Song et al. (2007) modeled the paper content by PLSA/

LDA topic model, while a number of studies (Zhang et al. 2010; Tan et al. 2006; Yang et al. 2006) tried to leverage external web information. With the metadata above, Han et al. (2004, 2003) built a Naive Bayes model for estimating the probability that a record belongs to an author. Yin et al. (2007) weighted and combined direct and indirect similarity by SVM. Treeratpituk and Giles (2009) proposed a random forest based binary classifier which can facilitate feature selection. The experiment shows that only a few important features are sufficient for accurate AD. They also proposed an IDF like feature weighting strategy.

To enable AD on large-scale data, Byung-won and Lee (2007), Byung-won et al. (2005) proposed to partition records into small groups with an inexpensive distance measure first, and then use a more expensive distance measure to disambiguate the records in each group. Recently, a number of researchers also imported active learning (Wang et al. 2011; Culotta et al. 2007), crowdsourcing (Cheng et al. 2013) and user feedback (Culotta et al. 2007; Qian et al. 2011; Godoi et al. 2013) to improve AD accuracy, and such approaches usually require external resources or human efforts.

Although many approaches have been proposed for solving the author disambiguation problem, the majority of them did not consider the dynamic setting where the data scale keeps enlarging and author names are becoming more ambiguous. In recent years, a few incremental author disambiguation methods have been proposed. Treeratpituk (2012) has proposed an incremental DBSCAN method by extending the batch DBSCAN method. It first computes the ε - neighborhood of a new record, which refers to a set of existing records whose distance to the new record is less than ε. If the neighborhood is dense (with more than *minPts* records) and contains records which belong to existing clusters, the new record will be assigned to existing clusters; otherwise, it will be assigned to a new cluster. Carvalho et al. (2011) proposed to insert new records into a cleaned digital library according to a number of pre-defined heuristics for checking whether new records belong to pre-existing clusters or new ones. Esperidião et al. (2014) enhanced the work of Carvalho et al. by dropping the assumption that the existing digital library is clean and by trying to merge incorrectly fragmented record clusters. They also proposed five record selection strategies for improving cluster purity. The CEN strategy, which compares the new record with only the representative records closer to the centroid of each cluster, is reported with the best overall performance in their experiments.

The above incremental AD methods have a few shortcomings. First, they rely on fixed parameters (Treeratpituk 2012) or a fixed set of heuristics (Carvalho et al. 2011; Esperidião et al. 2014) in the incremental AD process, which may not be suitable anymore after a number of updates. Second, when considering whether a new record should be assigned to an existing cluster, the existing methods compare the new record with the individual records in the cluster. Since this process may often involve a large fraction of existing records, it is computationally expensive.

In this paper, we propose a novel "BatchAD+IncAD" framework for dynamic author disambiguation. Our BatchAD algorithm takes an unsupervised approach, since we cannot assume any prior knowledge about authors at the beginning, and the choice of BatchAD will be further discussed in Sect. 4.3.1. On the other hand, our IncAD algorithm takes the multi-classification approach, since IncAD is used after BatchAD, that is, after some knowledge of disambiguated author names have been accumulated. But besides assigning new records to some existing record clusters, IncAD could also establish new clusters or merge existing clusters. In our experiments, we consider a few of the aforementioned batch methods as alternatives to BatchAD: *k*-means (Han et al. 2003), DBSCAN (Huang et al. 2006), DISTINCT (Yin et al. 2007), CollectiveER (Collective Entity Resolution) (Bhattacharya and Getoor 2007) and ProbModel (Probabilistic Model) (Tang et al. 2012). We will also compare

IncAD with the incremental author disambiguation methods discussed above, namely Treeratpituk's method (2012) and Esperidião et al.'s method (2014). As Esperidião et al.'s method (2014) is an enhancement of Carvalho et al.'s method (2011), we did not implement the latter as a baseline.

4 Our method

4.1 Overview

Figure 5 illustrates our AD framework, which comprises BatchAD for processing existing data in one go, and IncAD for repeatedly processing updates and for correcting existing results if necessary. To facilitate such periodical and frequent updates, an index structure for AD results is also built. Sections 4.2–4.4 describe our indexing, BatchAD and IncAD steps, respectively.

First, we would like to introduce some definitions:

- Author name: a string such as "Lei Zhang".
- Record: a "paper - author name" pair such as r_i: (<paper1>, <"Lei Zhang">), in which "Lei Zhang" is listed as one author of paper1.
- A set of records: $R = \{r_1, r_2, \ldots, r_n\}$ with the same author name (including name variations) is often the input of an AD algorithm.
- A set of disjoint record clusters: $A = \{A_1, A_2, \ldots, A_{k'}\}$ such that $A_1 \cup A_2 \cup \ldots \cup A_{k'} = R$ (that is, A completely covers R). In general, the output of an AD algorithm can be represented by A, where k' is the number of record clusters in A.
  In particular, let the gold set of record clusters be $A^* = \{A_1^*, A_2^*, \ldots, A_k^*\}$, where there is a one-to-one mapping between $A_i^*(i \in [1, k])$ and real-world authors. That is, A^* represents the correct partitioning of R, and k is the correct number of real-world authors that are covered by R.

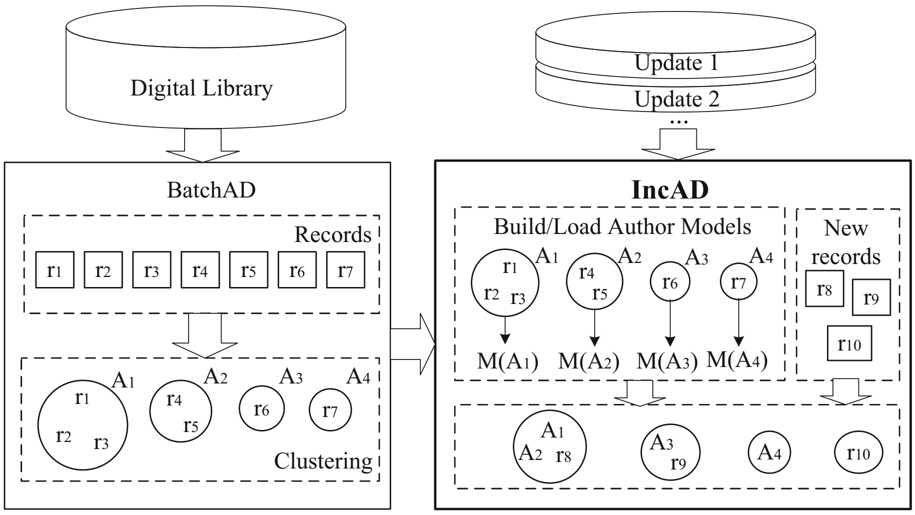

Fig. 5 Dynamic AD framework

According to the above notations, we now formalize the tasks of BatchAD and IncAD as follows:

BatchAD: *Given* $R = \{r_1, r_2, \ldots, r_n\}$, *try to find the correct partitioning* $A^* = \{A_1^*, A_2^*, \ldots, A_k^*\}$. *In practice, because the BatchAD algorithm is not perfect and k is unknown, it outputs* $A = \{A_1, A_2, \ldots, A_{k'}\}$.

IncAD: *Given* $A = \{A_1, A_2, \ldots, A_{k'}\}$ *and a new record* r_{new}, *either 1) create a new cluster that contains only* r_{new} *(i.e., the author name of* r_{new}*probably refers to a new author); 2) assign* r_{new} *to* A_i*for a particular* $i(1 \le i \le k')$; *or 3) merge two or more existing record clusters in A and assign* r_{new} *to the merged cluster.*

4.2 Indexing

Before AD, we need to decompose each paper into one or more records according to the number of author names it contains. For BatchAD, we need to divide all records in a digital library into smaller candidate record sets, so that the records in each candidate set are with the same author name (or their variations). Also, with every periodical update of IncAD, the data need to be loaded and stored many times. We do not know in advance which author names' AD results will be loaded and updated, and in what order. It is therefore essential to design an efficient indexing system. For those purposes, we have designed an index structure as shown in Fig. 6. The index first sorts records according to the last names of the author names in alphabetical order; each last name further points to a group of candidate record sets.

We take the following steps to build the index from scratch:

Step 1 Enumerate all author names in all records;
Step 2 Construct a dictionary of last names and group records by their last names;

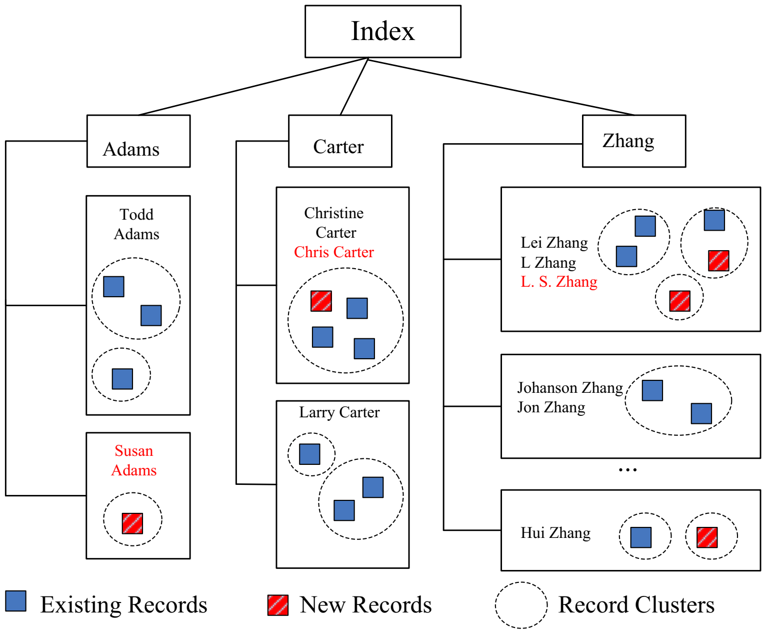

Fig. 6 The indexing structure of AD results

Table 2 Four types of name variations

Name variation case	Examples
Abbreviations	"L Zhang" & "Lei Zhang"
Nick names	"Chris Carter" & "Christine Carter", "Jon Zhang" & "Johanson Zhang"
Middle names	"L. S. Zhang" & "L Zhang"
Hyphenated names	"Z.Q. Nie" & "Z Q Nie"

Step 3 Further divide each group by building minimum connected graphs based on the author name similarity (Rahm and Bernstein 2001). Four types of name variations are regarded as similar names with a full name: namely, abbreviated names, replacement of nick names, adding middle names and using hyphenated names, as illustrated in Table 2. For recognizing nick names, we utilized an English nick name dictionary.[1]

In IncAD, when a new record arrives, we will try to add it to an existing candidate record set according to the author name similarity. If such set is not available, we will create a new candidate set for the new record: see "Susan Adams" in Fig. 6.

The benefits of building the index are as follows. First, because the author name similarity approach is relatively robust to author name variants, this is good for high AD recall (i.e. ensuring that records of the same authors are put together). Second, after the dividing process, the overall AD computational cost is reduced drastically, as it narrows down the AD scope from the whole record set to each candidate record set. Furthermore, the divided candidate sets could be further processed in parallel on multi-threads or multiple machines. Finally, in the IncAD phase, the algorithm only needs to load previous AD results of required candidate set onto memory.

In the following sections, we will describe how BatchAD and IncAD work given one candidate record set.

4.3 BatchAD

4.3.1 Choice of BatchAD

As BatchAD may need to disambiguate highly ambiguous author names which refer to large candidate record sets, it should be efficient and accurate.

Since we cannot assume any prior knowledge such as how many authors exist for a candidate record set, only unsupervised approaches are applicable. Typical unsupervised methods such as density based clustering, spectral clustering and graph partition require a similarity matrix as input which is expensive to construct (especially for highly ambiguous names), because it requires each and every pair of records to be compared. Thus, our BatchAD modifies a conventional clustering algorithm to ensure efficiency.

For accuracy, collective entity resolution works best when ambiguous author names co-occur in a paper. As an approximation, we disambiguate names in smaller candidate sets

[1] http://usefulenglish.ru/vocabulary/english-names.

first and then move on to bigger candidate sets, as the small ones usually refer to less-ambiguous names, and the resolved names might be of benefit to subsequent candidate sets. Probabilistic models enhance the AD accuracy by considering multiple record relations in a graph model together, but solving the model can also be costly. For simplicity, we encode the multiple relations in the similarity function, i.e. coauthor similarity, citation similarity and venue similarity.

In all, our BatchAD contains two components: multiple metadata features for record similarity calculation, and a modified conventional clustering algorithm for efficient AD.

4.3.2 Similarity calculation

We first describe how we compute $S(r_i, r_j)$, the similarity between two records that have a common author name.

Since each record contains multiple and incomplete metadata, we use the metadata shown in Table 3 for computing the similarity. As can be seen, we use two author related features, five paper related features and two link related features. Those metadata are denoted as F_1 to F_9 respectively. We did not use features such as the full paper text to ensure efficient processing.

We represent each metadata as a vector of items shown in Eq. 1. The metadata "Author name (F_1)" "Publication year (F_7)" are two exceptions, as will be described later.

$$\overrightarrow{F_h} = (a_{h1}, a_{h2}, a_{h3}, \cdots) \ . \tag{1}$$

For example, in $\overrightarrow{F_2}$ (*Coauthor name*), each vector item is a coauthor name. If a coauthor name is already disambiguated, we use the resolved author identifier instead. For obtaining the vectors $\overrightarrow{F_3}$ (*Paper title*) and $\overrightarrow{F_4}$ (*Paper abstract*), word bigrams are generated from the original text after removing stop words.[2] We chose bigrams because unigrams may result in too many false matches, while higher n-grams will suffer from a data sparsity problem as the texts in paper titles and abstracts are often short.

For accuracy consideration, we further weight the aforementioned metadata items by what we call IPF (Inverse Paper Frequency). It is calculated based on the frequency $f(a_{hi})$ of a feature item a_{hi} in all paper set, and N_h is the maximum $f(a_{hi})$ of $\overrightarrow{F_h}$

$$IPF(a_{hi}) = w_{hi} = \log_2 \frac{N_h + 1}{f(a_{hi}) + 1} \ . \tag{2}$$

Hence, each metadata is further represented as a weighted vector:

$$\overrightarrow{F_h} = (a_{h1} : w_{h1},\ a_{h2} : w_{h2},\ a_{h3} : w_{h3},\ \cdots) \ . \tag{3}$$

Now, the similarity between two records in regards of $\overrightarrow{F_h}$ can be calculated as:

$$S_h(r_i, r_j) = \frac{\overrightarrow{F_h(r_i)} \cdot \overrightarrow{F_h(r_j)}}{|\overrightarrow{F_h(r_i)}||\overrightarrow{F_h(r_j)}|} \ . \tag{4}$$

[2] ftp://ftp.cs.cornell.edu/pub/smart/english.stop.

Table 3 Three groups of metadata of a record

Metadata type	Metadata
Author related	F_1 (*Author name*)
	F_2 (*Coauthor names*)
Paper related	F_3 (*Paper title*)
	F_4 (*Paper abstract*)
	F_5 (*Paper keywords*)
	F_6 (*Publication venue*)
	F_7 (*Publication year*)
Link related	F_8 (*Reference*)
	F_9 (*Download link*)

Besides, two record's author name similarity ($S_1(r_i, r_j)$) is regarded as "1.0" if they are identical, "0.8" if they are similar according to the cases in Table 2, and "0.0" otherwise.

Two record's publication year (F_7) similarity is calculated as follows:

$$S_7(r_i, r_j) = \frac{T_{max} - |F_7(r_i) - F_7(r_j)|}{T_{max}} \tag{5}$$

where T_{max} is the maximum publication year interval in all paper set.

Based on the above similarities, we define the similarity profile feature vector $\overrightarrow{X(r_i, r_j)}$ as

$$\overrightarrow{X(r_i, r_j)} = (S_1(r_i, r_j), S_2(r_i, r_j), \ldots, S_9(r_i, r_j)) \ . \tag{6}$$

Finally, the similarity between a pair of records is computed using trained parameters $\vec{w}$ and b, as follows:

$$S(r_i, r_j) = <\overrightarrow{X(r_i, r_j)}, \vec{w}> - b \ . \tag{7}$$

Here, $\vec{w}$ is a vector of weights, and b is a threshold. These parameters are trained with labeled author data (will be described in Sect. 5.1.1) using linear SVM (Joachims 1999), implemented in the SVMlight toolkit.[3] In the training phase, we tune the similarity function for high-precision, i.e. we judge that two records belong to the same author only when given ample evidence. As a result, $S(r_i, r_j) > 0$ suggests that the two records belong to the same author, and $S(r_i, r_j) \leq 0$ indicates otherwise.

4.3.3 *Efficient clustering*

Based on the similarity function as defined above, BatchAD performs hierarchical agglomerative clustering, which has been shown to be efficient and reliable (Song et al. 2007; Yin et al. 2007; Pereira et al. 2009; Monz and Weerkamp 2009; Torvik and Smalheiser 2008). As our similarity computation is highly precision oriented, we can leverage transitivity to avoid computing the similarity for every record pair. Algorithm 1 shows our efficient clustering algorithm: the records form a graph whose edges represent similarities of positive values.

[3] http://svmlight.joachims.org/.

Algorithm 1 BatchAD

```
Require: A set of records $R = \{r_1, r_2, \cdots, r_n\}$
Ensure: A partition of $R$: $A = \{A_1, A_2, ..., A_{k'}\}$
 1: $A = \emptyset$
 2: Initiate two queues: an unprocessed queue: $U = \{r_1, r_2, \cdots, r_n\}$; a processed
    queue: $Q = \emptyset$
 3: Pop one $r_i$ from $U$, push it in $Q$
 4: i=0
 5: while $i < |Q|$ do
 6:   for $j = 0$ to $|U|$ do
 7:     if $S(Q[i], U[j]) > 0$ then
 8:       Delete U[j] from U
 9:       Push U[j] in Q
10:     end if
11:   end for
12:   i=i+1
13: end while
14: $A = A \cup \{Q\}, Q = \emptyset$
15: if $U \neq \emptyset$ then
16:   Go to step 3
17: end if
```

If all the n records belong to the same cluster, all of them will be processed in the first iteration, so only one iteration and $n - 1$ comparisons are needed. If each record belongs to a different cluster, $n - 1$ iterations and n^2 comparisons are needed. In practice, when the n records belong to k' clusters, k' iterations and about $O(k'n)$ comparisons are needed. As $k' < n$, it is much more efficient than conventional methods that rely on the n^2 similarity matrix. Before clustering, we rank the records according to the citation count, because highly ranked records are likely to be more representative and more informative, so processing those records at the early stage can help to quickly establish good start exemplars so that the pairwise comparison could be reduced.

4.4 IncAD

We now describe our IncAD algorithm, which is designed to incrementally disambiguate the author names of new records, after BatchAD has processed a collection of records. First, we will discuss the possible choices for designing an IncAD algorithm. Then, we will describe our solution of how IncAD processes a single new record. Furthermore, we will discuss how IncAD can update the data periodically (e.g. every other day or every week) when a mini-batch of new records come in, which is a more practical setting.

4.4.1 Choices of IncAD

With all author names in a digital library disambiguated, we have three types of choices for disambiguating the author name of a new record r_{new} :

- *Solution 1*: Using the index described in Sect. 4.2, we could find all records that have the identical (or similar) author names with r_{new}, noted as $R = \{r_1, r_2, \ldots, r_n\}$. Then, Algorithm 1 could be applied to $\{r_{new}\} \cup R$ in order to get the new AD results. The complexity is $O(k'n)$.
- *Solution 2*: By not re-clustering the records in $R = \{r_1, r_2, \ldots, r_n\}$, one can try to merge r_{new} into R's AD results $A = \{A_1, A_2, ..., A_{k'}\}$ incrementally. The solutions proposed in Treeratpituk (2012), Carvalho et al. (2011), Esperidião et al. (2014) followed this choice. Their methods require the similarity between r_{new} and the records in each A_i ($i \in [1, k']$) to be computed. Since $A_1 \bigcup A_2 \bigcup \ldots \bigcup A_{k'} = R$, the complexity is O(n).
- *Solution 3*: As the number of records n might expand fast with the growth of the digital libraries, we try not to compare r_{new} with existing records one by one. Instead, we compare r_{new} with each A_i ($i \in [1, k']$) directly, so that the computational complexity could be reduced to O(k'). Since $O(k') < O(n) < O(k'n)$, our IncAD follows the third solution.

4.4.2 IncAD for a new record

In this section, we first study a basic problem in IncAD: given r_{new} and a record cluster A_i ($i \in [1, k']$), decide whether r_{new} should be assigned to A_i.

Since a record cluster A_i tends to map to a real-world author (although the mapping may not be as perfect as the gold clusters), we can try to model the author profile from the records. This is possible because we can find certain patterns from the records' metadata. For example, an author usually has a certain set of frequent coauthors, has certain research topics (with regard to the title, abstract and keyword metadata), and publishes papers on a certain set of venues.

Following this intuition, we propose to build an author model for each record cluster. Here, we note A_i's author model as $M(A_i)$. Recall that each record in A_i has nine metadata including author name (F_1), coauthor names (F_2), paper title (F_3), … and download link (F_9), so we propose to build nine metadata model $M_1(A_i)$, $M_2(A_i)$, …, $M_9(A_i)$ to represent the author model $M(A_i)$. Here, $M_1(A_i)$ is the author name model: it captures the information of "What name variations does the author often use?"; $M_2(A_i)$ is the coauthor name model: it captures the information of "which coauthors is the author likely to have?"; and $M_3(A_i)$ is the paper title model, it captures the information of "which word bigrams often appear in the author's paper title?"; etc.

To build $M_h(A_i)$ (except the publication year model), we try to estimate A_i's probability distribution over the items of metadata F_h. First, we list all occurrence of items of F_h in $R = \{r_1, r_2, \ldots, r_n\}$ to form a metadata vocabulary $V_h = \{a_{h1}, a_{h2}, a_{h3}, \ldots\}$. In addition, we add one more item $a_{h\triangle}$ into V_h to represent the items unseen so far. Then, we estimate A_i's probability distribution over V_h. Specifically, the task is to estimate $p(a_{hj}|M_h(A_i))$, the probability that A_i has item a_{hj}, so that:

$$\sum_{j=1}^{|V_h|} p(a_{hj}|M_h(A_i)) = 1. \tag{8}$$

Here, $p(a_{hj}|M_h(A_i))$ is estimated according to Eq. 9:

$$p(a_{hj}|M_h(A_i)) = \begin{cases} \lambda & if \;\; a_{hj} = a_{h\triangle} \\ (1-\lambda)\dfrac{w_{hj}C(a_{hj}) + w_{hj}}{\sum_{j=0}^{|V_h|} w_{hj}C(a_{hj}) + \sum_{j=0}^{|V_h|} w_{hj}} & otherwise \end{cases} . \tag{9}$$

In this equation, $C(a_{hj})$ is the occurrence count of a_{hj} in all records of A_i, and w_{hj} is the IPF weight of a_{hi} as demonstrated in Eq. 2.

Now with the arrival of r_{new}, we have observed r_{new}'s metadata items. Noting r_{new}'s item set of metadata F_h as $F_h(r_{new}) = \{a'_{h1}, a'_{h2}, a'_{h3}, \ldots\}$, the probability that $M_h(A_i)$ "produces" r_{new} can be computed as:

$$\begin{aligned} p\big(r_{new}|M_h(A_i)\big) &= p\Big(a'_{h1}, a'_{h2}, a'_{h3}, \ldots|M_h(A_i)\Big) \\ &= \prod_{j=1}^{|F_h(r_{new})|} p\big(a'_{hj}|M_h(A_i)\big). \end{aligned} \tag{10}$$

To ensure a conservative decision, if all of r_{new}'s items are regarded as $a_{h\triangle}$ (the unseen item), we directly set $p\big(r_{new}|M_h(A_i)\big) = 0$.

For $M_7(A_i)$ (the publication year model), we compare the publication year between r_{new} and the average of A_i's records:

$$p\big(r_{new}|M_7(A_i)\big) = \frac{T_{max} - \left| F_7(r_{new}) - \frac{\sum_{r_j \in A_i} F_7(r_j)}{|A_i|} \right|}{T_{max}}. \tag{11}$$

Now we aggregate each metadata model's prediction result to decide whether r_{new} should be assigned to A_i.

$$I(r_{new}, A_i) = f\Big(p\big(r_{new}|M_1(A_i)\big), p\big(r_{new}|M_2(A_i)\big), \ldots, p\big(r_{new}|M_9(A_i)\big)\Big) \tag{12}$$

The function f is trained with labeled data using SVM Joachims (1999). As a result, a positive value of $I(r_{new}, A_i)$ indicates that r_{new} should be assigned to A_i, and a negative value otherwise.

Thus, the cluster that r_{new} should be assigned to can be represented as Eq. 13:

$$A(r_{new}) = \{A_i | I(r_{new}, A_i) > 0, i \in [1, k']\} . \tag{13}$$

When $|A(r_{new})| = 0$, we should create a new cluster that contains only r_{new}, which indicates that the author name of r_{new} refers to a new author. When $|A(r_{new})| = 1$, we assign r_{new} to one of the existing clusters, as the author name probably represents one of the authors already known. When $|A(r_{new})| \geq 2$, it means two or more existing clusters should be merged, and r_{new} should be assigned to the merged cluster. In this case, r_{new} helps to fix some recall errors in previous AD results.

4.4.3 Periodical IncAD

In practice, new papers (and therefore new records) often arrive periodically in small batches. This happens, for example, when a new journal issue or a conference proceeding is registered to the digital library. IncAD handles such updates by periodically running three steps, namely: data loading, new record assignment and author model update.

1. *Data Loading*: In an update, we first list all author names in the newly arrived records, and try to find for each name a matching record candidate set from the index described in Sect. 4.2. If no such candidate set is found for an author name, we create a new one for it. For an author name with an existing candidate set, we load the set's containing records and their AD results (i.e. record clusters) onto memory. For each record cluster, we build its author models. Actually, this only needs to be done when IncAD is invoked for the first time. For subsequent IncADs, we can load the author models directly without loading the individual records.
2. *New Records Assignment*: Following the method described in the last subsection, we could assign each new record to existing record clusters in three ways, as Figure 7 illustrates. Note that for all new records which do not belong to any existing clusters, we further apply Algorithm 1 to group them into new clusters.
3. *Author Model Update*: With the assignment of new records, we further update all author models. The procedure is: (1) update the vocabulary V_h by adding new

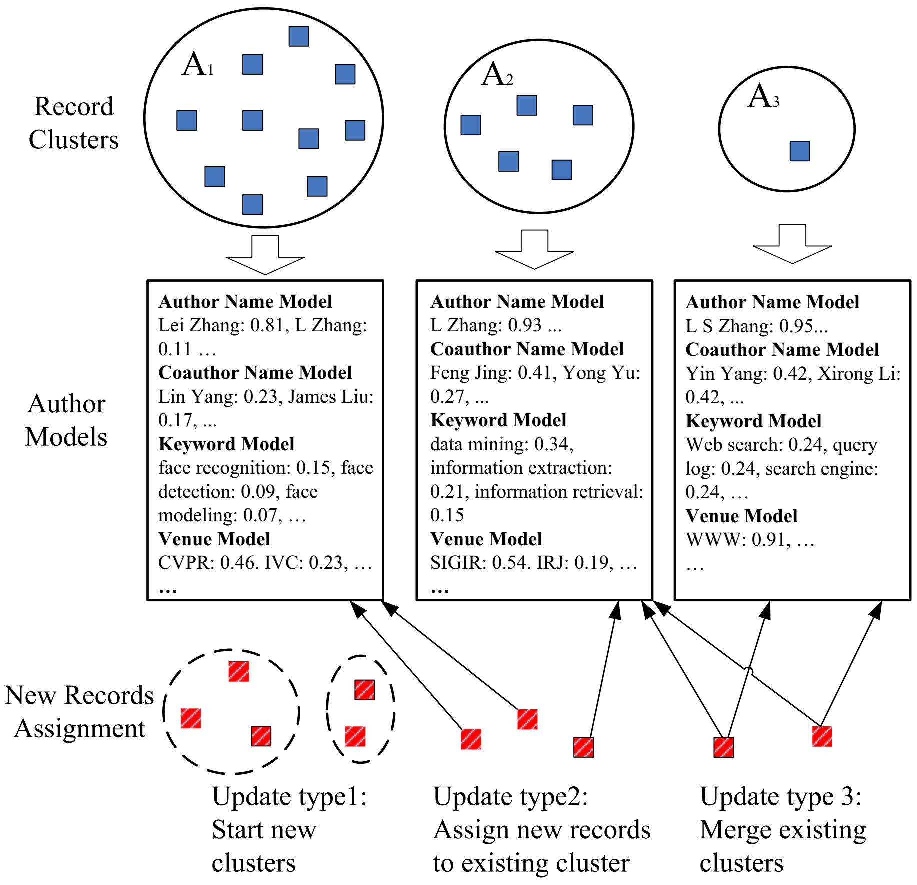
Record Clusters
A1
A2
A3
Author Models
Author Name Model
Lei Zhang: 0.81, L Zhang: 0.11 …
Coauthor Name Model
Lin Yang: 0.23, James Liu: 0.17, ...
Keyword Model
face recognition: 0.15, face detection: 0.09, face modeling: 0.07, …
Venue Model
CVPR: 0.46. IVC: 0.23, …
…
Author Name Model
L Zhang: 0.93 ...
Coauthor Name Model
Feng Jing: 0.41, Yong Yu: 0.27, ...
Keyword Model
data mining: 0.34, information extraction: 0.21, information retrieval: 0.15
Venue Model
SIGIR: 0.54. IRJ: 0.19, …
…
Author Name Model
L S Zhang: 0.95...
Coauthor Name Model
Yin Yang: 0.42, Xirong Li: 0.42, ...
Keyword Model
Web search: 0.24, query log: 0.24, search engine: 0.24, …
Venue Model
WWW: 0.91, …
…
New Records Assignment
Update type1: Start new clusters
Update type2: Assign new records to existing cluster
Update type 3: Merge existing clusters

Fig. 7 An example of using author models for IncAD

Table 4 Statistical information of two labeled data sets

Statistics	CaseStudy	DBLP
# of records	1322	6,783
# of author names	54	672
# of author labels	171	1,201
Avg/Max labels per name	3.96/117	1.79/17
Avg/Max names per label	1.25/4	1.00/2
Avg/Max records per label	7.73/323	5.65/125

metadata items in newly arrived records; (2) for each newly added record cluster, build its author models from scratch; (3) for each existing cluster with new records added, update its author model by updating its distribution over the metadata vocabulary; (4) for each merged cluster, build its author model by merging each cluster's distribution over the metadata vocabulary. After this procedure, all updated record clusters and author models are stored with the index structure as Fig. 6 shows, in order to facilitate future updates.

5 Experiments

Section 5.1 describes our experimental setups including data sets and baselines. Section 5.2 evaluates the efficiency of our BatchAD and IncAD algorithms compared with baselines. Section 5.3 evaluates the accuracy of BatchAD, IncAD, baselines, as well as different batch/incremental combinations. Finally, Sect. 5.4 provides an additional analysis on IncAD (Table 4).

5.1 Experimental setup

5.1.1 Data sets

In our experiments, we use two labeled data sets, *CaseStudy* and *DBLP*, in which the true author label of each record is available, and one large unlabeled data set, *Unlabeled*, which is composed of raw data without true author labels. We use *CaseStudy* for training $\overrightarrow{w}$ and b in Eq. 7, and for training the function f in Eq. 12. We use *Unlabeled* for the efficiency evaluation, and *DBLP* for the accuracy evaluation.

All data were obtained via the Microsoft Academic Search API,[4] which provided a rich set of metadata including all the nine metadata discussed in Sect. 4.3.2. While the API also provided its own AD information such as (estimated) author affiliation and research interests, we did not utilize such data, as we want our algorithms to be independent of the results from other AD systems. Moreover, the API provided a ranked list of authors, papers, keywords and conferences, based on their occurrence counts in all paper set. We used the statistics to compute IPF (see Eq. 2).

The *CaseStudy* dataset, our training data set, was originally created for our pilot study. This data set contains 1322 records in the computer science discipline for 54 unique author

[4] http://academic.research.microsoft.com

names (which in fact represent 171 different authors). Note also that an author may have name variations due to abbreviations etc. We hired five graduate students to manually assign a true author label to each record by referring to external resources such as the authors' contact information, webpages and by contacting the authors by emails where necessary. Specifically, each record was labeled by two students independently, and inconsistencies were solved through discussions. This dataset contains highly ambiguous author names, such as 117 different "Lei Zhang"(including the variations of L. Zhang and L. S. Zhang).

The *DBLP* dataset, our test data set, was an aggregation of several public available data sets. Several previous studies (Han et al. 2004; Byung-won et al. 2005; Wang et al. 2011) selected a data sample from DBLP (Digital Bibliography & Library Project) and manually created author disambiguation test sets from it. We collected all available data with author labels including 9160 records. Due to some inconsistencies among the data, we hired 10 labelers to check the correctness of author labels under the same criteria used to label *CaseStudy*. We found that 1.23 % of original labels was incorrect. Finally, after removing records without any metadata and records whose author labels could not be clearly identified, we obtained 6783 records. To facilitate future research in the community, we have made this cleaned dataset available online.[5]

The *Unlabeled* dataset, a raw dataset including 3.5 million papers, was crawled by using ten common person names as seeds, as Table 5 shows. At each step, we used the Microsoft Academic Search API to first retrieve papers by those seed person names, and then expanded our search by utilizing the current papers' coauthors and references. Figure 8 shows the cumulative distribution of the publication year metadata for the crawled papers. The figure also exemplifies the exponential growth of digital libraries.

5.1.2 Baselines

For BatchAD, we chose five state-of-the-art batch AD methods as baselines, namely: *k-means* (Han et al. 2003), *DBSCAN* (Huang et al. 2006), *DISTINCT* (Yin et al. 2007), *CollectiveER* (Bhattacharya and Getoor 2007), and *ProbModel* (Tang et al. 2012). For *k-means*, *k* was set to the number of actual authors: thus the results obtained in our experiments were optimistic. The original implementation of each baseline utilized different subsets of metadata. Here, we utilized all metadata in Table 3 for each of them.

For IncAD, we adopted three incremental AD methods as baselines. The first baseline applies the clustering method in Algorithm 1 to disambiguate the author names in newly arrived records, so we note it as *IncClustering*. The second baseline is an incremental version of *DBSCAN* discussed in Treeratpituk (2012), and we note it as *IncDBSCAN*. The third baseline is proposed in Esperidião et al. (2014), noted as *IncHeuristic*. It adopts predefined heuristics to insert new records into existing results. In the experiments, we adopted best record selection strategy CEN as the authors reported. It compares a new record with only the records closer to the centroid of each cluster.

We implemented each baseline method as faithfully as possible by reading the original papers, and tuned the required parameters with the *CaseStudy* dataset. Note that each baseline method was originally evaluated on a different dataset from different sources. In order to enable a fair comparison, we uniformly used *Unlabeled* and *DBLP* to evaluate the efficiency and accuracy for each of them.

[5] https://github.com/yaya213/DBLP-Name-Disambiguation-Dataset.

Table 5 The statistical information of the *unlabeled* dataset

# of names	510,163
# of papers	3,530,791
# of records	11,276,739
Seed Names	Y. Yang; Wei Zhang; Y. Suzuki; Jun Wang; V. Singh; Paul Thompson; Lei Zhang; K. Mori; Wen Xu; Mario Gerla

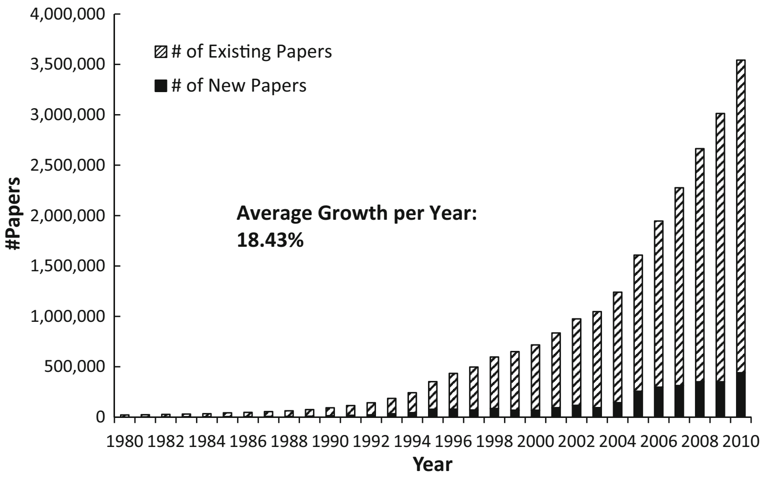

Fig. 8 Cumulative paper count distribution for the *unlabeled* dataset

5.2 Efficiency

In this section, we will evaluate the efficiency of BatchAD, IncAD and baselines according to each method's running time cost on the *Unlabeled* dataset. First, we will compare the efficiency between BatchAD and BatchAD+IncAD; then, we will analyze the IncAD efficiency with different update periods; further, we will compare the efficiency between IncAD and three baselines; in addition, we will also compare the efficiency between BatchAD and five baselines. Our experimental platform for efficiency evaluation is as follows:

- CPU: Intel Core i7-2600CPU @ 3.40GHz 8 cores
- Memory: 8.00GB
- Operating System: Windows 7 x64
- Compiler: Visual Studio 2010, Microsoft .NET Framework 4

5.2.1 *BatchAD versus BatchAD+IncAD*

We compared the efficiency of BatchAD and BatchAD+IncAD using the *Unlabeled* dataset. When BatchAD was used to process the entire *Unlabeled* data, it cost 3 h and 20 min. Thus, BatchAD alone is not suitable for frequent periodical updates. As we reviewed in Sect. 2.1, new data are emerging at a fast rate, so digital libraries may need to update the AD results frequently, such as every other day or every week. In the BatchAD+IncAD setting, it took 4 min for BatchAD to process all records with the publication year in or before 2000 (which is about 20 % of all the records in *Unlabeled*). Then, IncAD was used repeatedly to process weekly updates (i.e. update period of 7 days). Since the metadata contain the published year information only but not the precise dates, we assigned a date to each paper at random.

Figure 9 shows the cost (i.e. time required by IncAD) of each weekly update. As the number of "known" record clusters and new records increases with time, IncAD's cost also grows. However, even when processing the very last weekly update, the cost of processing is only about 39 s, which is practically feasible.

5.2.2 *IncAD with different update periods*

For the BatchAD+IncAD approach, we also varied the update period from 1 to 30 days as shown in Table 6. Not surprisingly, the longer the update period is, the more time each update costs, since it needs to handle more data. However, the cost is not proportional to the period length. For example, the update period of 30 days is about 4 times as long as an update period of 7 days, so that each monthly update needs to handle about 4 times the new records of each weekly update. However, the former only costs about 30 % more time than the latter. To analyze this phenomenon, we further investigated the cost of each step in IncAD, namely: (1) data loading cost, (2) new records assignment cost, and (3) author model update cost. Table 6 shows the result.

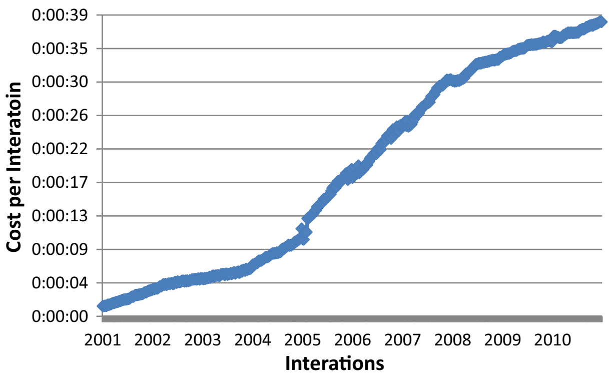

Fig. 9 IncAD's cost of per weekly update (#Interations = 522)

Table 6 IncAD cost of different update periods

Update period (days)	Cost per iteration (s)	Data loading cost (s)	New records assignment cost (s)	Author model update cost (s)
1	11.44	3.58	0.36	7.50
3	15.45	3.84	0.62	10.99
7	19.07	4.46	1.35	13.26
15	21.36	4.61	3.21	13.54
30	24.79	4.71	6.33	13.75

From the results, we found that only the “new records assignment” cost is proportional to the number of new records. The costs of “data loading” and that of “author model update” are similar across different update periods especially for the 7–30 days range. This means that, in those settings, even when different sizes of new data were added, a comparable number of author models were required to be loaded and updated. Another finding is that the cost of “data loading” and “author model update” are high even for frequent updates (1–3 days), and therefore such very short update periods are not economical. As long as the update delay and accuracy (which we will discuss in Sect. 5.3.5) are acceptable, a longer update period should probably be used.

5.2.3 *Efficiency comparison between IncAD and baselines*

We further compare IncAD efficiency with three baselines: *IncClustering*, *IncDBSCAN* and *IncHeuristic*. Table 7 compares them in terms of computational complexity. Here, n is the number of existing records and k' is the number of record clusters in the existing AD results. Since $O(k') < O(n) < O(k'n)$, IncAD is more efficient than the three baselines.

Moreover, Fig. 10 compares the actual costs between IncAD and the three baselines. *IncClustering* is the most time-consuming method. It took more than 2 h to complete the very last weekly update on *Unlabeled*. This cost is even comparable with that of processing the whole dataset with BatchAD (which is 3 h and 20 min), so *IncClustering* is not feasible for updating large-scale data frequently. This is because *IncClustering* needs to re-cluster all existing records whose author names coincidely overlap with those in the new records, and such existing records often occupy a large fraction. *IncDBSCAN* does not re-compute existing record clusters, but it needs to compare each new record with every existing record with the same author name. In the very last update, it costs about 2 min, which is 318 % of

Table 7 Complexity comparison between IncAD and three baseline methods

Method	Complexity
IncClustering	$O(k'n)$
IncDBSCAN	$O(n)$
IncHeuristic	$O(n)$
Our method (IncAD)	$O(k')$

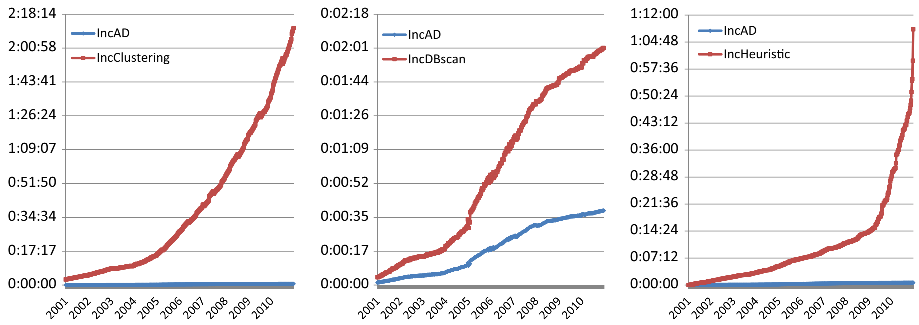

Fig. 10 Cost comparison between IncAD and three baselines in the weekly update setting

the time used by our method. The cost of *IncHeuristic* grows quickly with the accumulate of more records. The major reason is the record selection strategy requires the centroid of each cluster to be re-computed and the representative records to be re-selected after each update. In conclusion, our IncAD method is the most computationally inexpensive one: it only costs 31.45 % time of the best baseline (IncDBSCAN) in the last weekly update on *Unlabeled*.

5.2.4 Efficiency comparison between BatchAD and baselines

Here, we compare BatchAD with five baselines in terms of efficiency. Table 8 shows the complexity of each method. *k-means* is often very time-consuming as it requires multiple iterations. In practice, the number of iterations m is unpredictable and often large. *DBSCAN* alone has a complexity of $O(n \log n)$, but it requires the construction of an n-by-n record similarity matrix as a priori (which is $O(n^2)$ complexity). *DISTINCT* requires frequent re-computation of cluster similarity for merging hierarchical clusters. *CollectiveER* needs a bootstrap stage and *ProbModel* needs to find atom clusters in preprocessing, whose complexity is comparable to our BatchAD method. Our BatchAD method is the most computationally-efficient one, requiring only $O(k'n)$ complexity.

We also compared the actual computational costs of these batch methods using the *Unlabeled* data set. With each method, we first processed all papers published before 2001 (see Fig. 11); then we processed all papers published before 2002, and so on. We terminated processes that ran over 12 h. We did not test the cost of *k-means* since k is unknown in *Unlabeled*. Figure 11 shows the result. It is clear that our BatchAD algorithm is the most

Table 8 Complexity comparison between BatchAD with five state-of-the-art batch methods

Method	Complexity
k-means	$O(knm)$
DBSCAN	$O(n^2) + O(n \log n)$
DISTINCT	$O(n^2 \log n)$
CollectiveER	$O(k'n) + O(n \log n)$
ProbModel	/
Our method (BatchAD)	$O(k'n)$

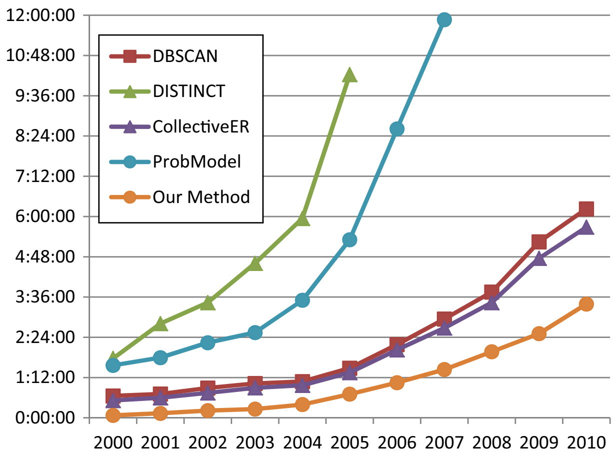

Fig. 11 BatchAD cost comparison between our method and baselines

computationally efficient one: it only costs 59.67 % of the time required by the best baseline (CollectiveER).

5.3 Accuracy

We now evaluate the accuracy of our methods. Here, *CaseStudy* was used for training and *DBLP* was used for testing, as these two data sets have gold-standard author labels.

In model training, we use the standard precision, recall and F1 measures to evaluate. (1) The similarity function in Eq. 7 is used as a binary classifier to judge whether two records belong to the same author or not. (2) The new record assignment function in Eq. 12 is used as a binary classifier to determine whether we should assign a new record to an existing record cluster or not.

For evaluating the AD results, which are represented as a clustering on a set of records, we use B-cubed precision, recall and F1 (Amigó et al. 2009) for evaluation. Recall that our system produces record clusters $A = \{A_1, A_2, ..., A_{k'}\}$ for a set of records, and the gold clustering of those records are $A^* = \{A_1^*, A_2^*, ..., A_k^*\}$. Suppose that a record r_i belongs to A_i in the produced record clusters, and it belongs to A_i^* in the correct clusters. B-cubed precision of r_i is the proportion of records in A_i which also belong to A_i^*. B-cubed recall of r_i is the proportion of records in A_i^* which also belong to A_i. The B-cubed F1 of r_i is a harmonic mean of its B-cubed precision and B-cubed recall. The overall B-cubed F1 is the averaged B-cubed F1 of all records. Since the average is calculated over records, it is not necessary to apply any weighting according to the size of the produced clusters or golden clusters.

5.3.1 *Model training*

(1) *Training the pairwise similarity function in BatchAD*

As we have mentioned earlier, *CaseStudy* was used for training $\overrightarrow{w}$ and b in Eq. 7 using linear kernel SVM. Each training sample is generated from a pair of records r_i and r_j sharing identical or similar author names. The feature vector of a training sample is $\overrightarrow{X(r_i, r_j)} = (S_1(r_i, r_j), S_2(r_i, r_j), \ldots, S_9(r_i, r_j))$ (Eq. 6). The label of a training sample is "+1" if the two records have the same author label, and "−1" otherwise. Then, the positive samples that lack shared metadata were filtered out (this can happen, for example, when the same author has two records in totally different research areas under different affiliations and with different coauthors). This filtering is done because we do not want the system to make a random guess when there is no overlap whatsoever in the metadata: we want it to make a conservative estimate.

Table 9 shows the accuracy of our pairwise similarity function as a binary classifier, both with the training data (*CaseStudy*) and with the test data (*DBLP*). It can be observed

Table 9 The accuracy of pairwise similarity prediction

Data Set	CaseStudy (train)	DBLP (test)
# of positive pairs	66,124	113,798
# of filtered pairs	34,436	40,201
# of negative pairs	84,327	180,017
Precision	98.65 %	97.02 %
Recall	98.01 %	94.91 %
F1	98.33 %	95.95 %

Table 10 The Accuracy of New Record Assignment

Data Set	CaseStudy (train)	DBLP (test)
# of "known" clusters	142	1,124
# of new records	398	2,257
# of cluster-record pairs	13,909	11,377
Precision	99.80 %	98.93 %
Recall	96.01 %	93.27 %
F1	97.86 %	96.02 %

that the similarity function accurately predicts whether a given pair of records have the same author label: the F1 for the test data is near 96 %; especially, the precision is promising.

(2) *Training the new record assignment function in IncAD*

CaseStudy was also used for training the assignment function f in Eq. 12, which determines whether a new record should be assigned to an existing record cluster in IncAD. In the training phase, we first randomly selected 70 % records and grouped them by the author labels to form record clusters (which represents "known authors"). Then, the remaining 30 % were regarded as new records. We paired each new record with all "known" clusters which have the identical (or similar) author name with it. From *CaseStudy*, we obtained 13,909 cluster-record pairs for training. For each training sample, we extracted its features according to the probability $p(r_{new}|M_h(A_i))$ $(1 \leq h \leq 9)$ introduced in Eqs. 10 and 11. As the absolute value of each probability is also affected by its vocabulary size $|V_h|$ and the number of unseen items in r_{new} (noted as $C_{\triangle h}$), we also take $|V_h|$ and $C_{\triangle h}$ as features from each metadata (except publication year). With the nine probability values, plus the additional eight vocabulary sizes and eight unseen item counts as features, we obtained 25 features altogether. Further, if the record's author label matches the cluster, the sample's ground truth is set as "1", otherwise "-1". Table 10 shows the results with a promising training F1 of 97.86 % and testing F1 of 96.02 %.

Using our training data (CaseStudy), we also tuned the parameter λ in Eq. 9 by setting it in the range from 0.001 to 0.05, with the interval of 0.001. The best performance was achieved when $\lambda = 0.012$.

5.3.2 BatchAD versus BatchAD+IncAD

Now we evaluate BatchAD+IncAD accuracy on both *CaseStudy* (training data) and *DBLP* (test data) with comparison to BatchAD. For BatchAD+IncAD, we first apply BatchAD to process all records in or before the year 2000 as input, and then use IncAD periodically to process weekly updates. In Table 11, we report the accuracy of BatchAD and BatchAD+IncAD for ten years (from 2001 to 2010). For example, for the year 2001, our BatchAD-only approach processed the entire data in and before 2001 in one go; as for BatchAD+IncAD, BatchAD was used to process the records in and before 2000, and then IncAD processed the records from 2001 as 52 weekly updates (where each update contains about 11.5 records on average in *DBLP*).

Table 11 compares the two approaches in terms of B-cubed F1, for weekly updates. With *DBLP* (test data), after processing the entire data set, the B-cubed F1 of BatchAD-only is 86.83 %, while that of BatchAD+IncAD is 84.25 %. Thus, there is no more than a 3 % drop in accuracy. The major reason for the drop is that while BatchAD-only makes use

Table 11 Accuracy comparison between BatchAD and IncAD (with update period = 7 days [a])

Year[a]	#Records	CaseStudy BatchAD F1 (B-cubed) (%)	BatchAD+IncAD F1 (B-cubed)[b] (%)	#Records	DBLP BatchAD F1 (B-cubed) (%)	BatchAD+IncAD F1 (B-cubed)[b] (%)
2001	238	92.42 ± 2.84	91.97 ± 2.44 $^-$	915	89.65 ± 4.86	89.87 ± 7.83
2002	292	92.67 ± 6.81	91.35 ± 5.23$^-$	1149	89.23 ± 6.27	89.31 ± 9.71
2003	366	93.53 ± 2.64	91.55 ± 3.23*	1527	88.80 ± 7.12	89.14 ± 2.78
2004	531	93.55 ± 3.48	90.73 ± 4.13*	2043	88.32 ± 4.06	88.41 ± 6.88
2005	699	93.19 ± 4.97	89.98 ± 5.03*	2755	88.92 ± 5.21	88.12 ± 2.75
2006	849	92.02 ± 5.37	88.78 ± 4.92*	3602	88.13 ± 4.97	87.53 ± 9.91
2007	989	89.45 ± 4.85	87.15 ± 3.73*	4459	87.26 ± 4.19	86.10 ± 7.16
2008	1146	88.59 ± 5.47	85.79 ± 3.63*	5411	87.26 ± 3.46	85.57 ± 1.20
2009	1326	86.48 ± 6.13	84.80 ± 4.15*	6332	86.76 ± 4.78	84.94 ± 1.59
2010	1332	86.28 ± 6.10	84.34 ± 5.32*	6783	86.83 ± 5.06	84.25 ± 6.83

[a] We only report the accuracy after each year here

[b] We conduct two-tailed paired t-test over each pair of BatchAD F1 and BatchAD+IncAD F1. "$*$" means the difference is significant with p value $<$ 0.05, "-" means the difference is not significant with p value ≥ 0.05

of the entire data set, BatchAD+IncAD first batch-processes the papers published in and before 2000 and then moves to the IncAD step to periodically process new updates. At a certain point, the available information is limited within existing records, while that of newly arrived records is not utilized yet. Especially, for a long update period, when the author models are not updated in time, the accuracy will be affected. In Sect. 5.3.5, we will discuss how different update periods affect the accuracy in more detail. However, considering the significant efficiency improvement of BatchAD+IncAD that we demonstrated in Sect. 5.2, this little sacrifice is clearly worthwhile from a practical point of view.

Table 11 also shows that, as the data size increases exponentially, the problem of AD becomes harder to solve. Nevertheless, even the final B-cubed F1 of BatchAD+IncAD (84.25 %) seems promising. Below, we compare our methods with existing methods in terms of accuracy.

5.3.3 BatchAD versus baselines

Next, we compare our proposed BatchAD with five state-of-the-art batch methods. Using *DBLP* as the test data, Table 12 shows the results in terms of accuracy. As indicated in the table, BatchAD statistically significantly outperforms the first four methods and is comparable to *ProbModel* in terms of B-cubed F1. *CollectiveER* owns the best precision since it only uses resolved coauthors as evidence. In its tuning process with the training data, we also found that its precision dropped quickly when trying to improve recall. Overall, since our method is the most computationally-cheap one as Sect. 5.2.4 reports, it is a promising choice considering both efficiency and accuracy.

Table 12 Accuracy comparison between BatchAD and five state-of-the-art methods

Method	k-means (%)	DBSCAN (%)	DISTINCT (%)	CollectiveER (%)	ProbModel (%)	BatchAD (%)
B-cubed precision	78.12	87.54	84.92	89.12	86.99	87.30
B-cubed recall	57.38	81.11	85.13	83.02	87.28	86.37
B-cubed F1[a]	$66.16^{\dagger}$	84.20^{*}	85.02^{*}	85.96^{*}	87.13^{-}	86.83

[a] We conduct two-tailed paired t test between the B-cubed F1 of BatchAD and the other five methods. "$*$" means the difference is significant with p value <0.05, "†" means p value <0.01

5.3.4 IncAD versus baselines

Further, we evaluate the IncAD accuracy on *DBLP* (test data) comparing with three baselines: IncDBSCAN, IncHeuristic and IncClustering.

In this experiment, we first apply BatchAD to process all records in or before the year 2000 as input, and then use the incremental AD methods repeatedly to process weekly updates. In Table 13, we report the accuracy of each method for ten years (from 2001 to 2010). From the results, we found that our method significantly outperforms BatchAD+IncDBSCAN after each year. Besides, our method has a comparable accuracy with BatchAD+IncHeuristic in the first 2 years (i.e. 104 weekly updates), and significantly outperforms it afterwards. The reason for the improvement is that IncDBSCAN and IncHeuristic regards an author (represented by a cluster of records) as a collection of individual records, while our method builds an author model to capture an author's profile from multiple dimensions of those records. Moreover, our author models are updated after each iteration, while the parameters in IncDBSCAN and the heuristics in IncHeuristic are fixed along the way. Although IncClustering owns exactly the same accuracy with BatchAD, it is too time-consuming and therefore not practical, as we have demonstrated in Sect. 5.2.3.

5.3.5 IncAD accuracy with different update periods

We also tested IncAD with different update periods (1–30 days) to investigate the effect of update frequency on accuracy.

Table 14 shows the B-cubed F1 values after processing the entire DBLP data set for different update periods. It can be observed that for IncAD, the shorter the update period, the more accurate the final result is. With the update period of 1 day, the final BatchAD+IncAD accuracy is very close to that of BatchAD (86.83 %). This is because frequent updates will provide the IncAD algorithm with more evidence early on. However, as we have discussed in Sect. 5.2.2, updating too frequently is computationally not economical, so an update period of 7 days or longer may be a practical choice.

5.3.6 Combining different batch methods with IncAD

Now we apply IncAD on top of different batch AD methods and evaluate the accuracy. First, we processed all DBLP data published in or before 2000 using one of the batch methods, and then processed the weekly updates from 2001 to 2010 using IncAD. As Table 15 shows, after 2003, the B-cubed F1 values of "DBSCAN+IncAD", "DISTINCT+IncAD" and

Table 13 Accuracy comparison between IncAD and baselines

Year[a]	# Records (%)	BatchAD + IncAD (%)	BatchAD + IncDBSCAN (%)	BatchAD + IncHeuristic (%)	BatchAD + IncClustering [b] (%)
2001	915	89.87 ± 7.83	86.44 ± 7.96 *	89.93 ± 2.37 −	89.65 ± 4.86 −
2002	1149	89.31 ± 9.71	85.28 ± 7.84 *	88.42 ± 3.79 −	89.23 ± 6.27 −
2003	1527	89.14 ± 2.78	84.08 ± 7.93 *	86.58 ± 4.97 *	88.80 ± 7.12 −
2004	2043	88.41 ± 6.88	83.11 ± 3.02 *	84.10 ± 3.43 *	88.32 ± 4.06 −
2005	2755	88.12 ± 2.75	81.21 ± 1.94 *	82.58 ± 7.24 *	88.92 ± 5.21 −
2006	3602	87.53 ± 9.91	81.00 ± 4.92% *	81.65 ± 1.23 *	88.13 ± 4.97% −
2007	4459	86.10 ± 7.16	80.52 ± 6.41 *	80.82 ± 3.70 *	87.26 ± 4.19 −
2008	5411	85.57 ± 1.20	79.78 ± 8.96 *	79.93 ± 2.56 *	87.26 ± 3.46 *
2009	6332	84.94 ± 1.59	79.39 ± 5.86 *	79.45 ± 2.56 *	86.76 ± 4.78 *
2010	6783	84.25 ± 6.83	79.32 ± 2.26 *	78.88 ± 1.03 *	86.83 ± 5.06 *

[a]The update period in this table is 7 days. We only report the accuracy after each year here

[b] The accuracy of BatchAD + IncClustering exactly equals that of BatchAD

[c] We conduct two-tailed paired *t* test between the B-cubed F1 of our method and the other two baselines. "$*$" means the difference is significant with *p* value <0.05, "-" means the difference is not significant with *p* value ≥ 0.05

Table 14 Accuracy Comparison between Different Update Periods

Update period (day)[a]	BatchAD + IncAD (%)
1	85.98 ± 1.69
3	85.12 ± 2.71
7	84.25 ± 6.48
15	83.87 ± 9.44
30	82.57 ± 8.26

[a] We only report the accuracy after the very last update here

Table 15 IncAD based on different batch methods (with update period = 7 days)

B-cubed	*k*-means	DBSCAN	DISTINCT	CollectiveER	ProbModel	BatchAD
F1	+ IncAD (%)	+ IncAD (%)	+ IncAD (%)	+ IncAD (%)	+ IncAD (%)	+ IncAD (%)
Batch	$71.15^{\dagger}$	$83.99^{\dagger}$	86.20^{*}	87.73^{*}	89.83^{-}	89.76
2001	$77.17^{\dagger}$	$86.04^{\dagger}$	86.52^{*}	87.89^{*}	89.95^{-}	89.87
2002	$80.36^{\dagger}$	86.94^{*}	87.85^{*}	88.00^{*}	88.91^{-}	89.31
2003	$82.87^{\dagger}$	87.70^{*}	88.26^{*}	88.29^{*}	89.29^{-}	89.14
2004	$82.00^{\dagger}$	88.51^{-}	88.42^{-}	88.84^{-}	88.00^{-}	88.41
2005	$81.26^{\dagger}$	87.23^{-}	87.50^{-}	87.74^{-}	87.82^{-}	88.12
2006	$81.50^{\dagger}$	86.22^{-}	86.68^{-}	86.99^{-}	86.58^{-}	87.53
2007	$81.80^{\dagger}$	85.99^{-}	85.97^{-}	86.58^{-}	85.65^{-}	86.10
2008	$81.66^{\dagger}$	84.85^{-}	84.60^{-}	84.93^{-}	84.94^{-}	85.57
2009	$81.91^{\dagger}$	84.65^{-}	84.18^{-}	83.94^{-}	84.67^{-}	84.94
2010	$81.03^{\dagger}$	84.09^{-}	84.31^{-}	83.71^{-}	84.53^{-}	84.25

[a] We conduct two-tailed paired *t* test between our method and the other five methods. "$*$" means the difference is significant with *p* value <0.05, "†" means *p* value <0.01, "-" means the difference is not significant

"CollectiveER+IncAD" are statistically indistinguishable from our proposed method "BatchAD+IncAD". Thus, IncAD works well with these batch methods as well. Moreover, note that "BatchAD+ IncAD" statistically significantly outperforms the first four methods until 2003. In other words, "BatchAD+IncAD" is quicker to achieve high performance than the other methods. As for "*k*-means+IncAD", it underperforms "BatchAD+IncAD" in every setting. However, it can also be observed that IncAD improves on *k*-means dramatically: compare the Batch row and the 2010 row. "ProbModel+IncAD" has the highest batch accuracy, but our method is comparable to it in all years.

5.3.7 Combining inaccurate BatchAD with IncAD

Finally, we study how IncAD performs given an inaccurate BatchAD method to start with. To this end, we created low-recall and low-precision versions of BatchAD, by setting b in Eq. 7 to a very high value and a very low value respectively. Figure 12 shows how IncAD performs with the low-recall and low-precision BatchADs. It is clear that while IncAD successfully improves on the low-recall BatchAD, it fails to do so for the low-precision BatchAD. This result shows that it is important for a batch AD algorithm to maintain high precision rather than high recall. Indeed, batch AD systems are precision-oriented rather than recall-oriented in practice, so our approach of BatchAD+IncAD is practical.

5.4 Further analysis

As we have discussed, when a new record is added, three types of updates may occur. In our IncAD experiment with *Unlabeled* (Sect. 5.2.1), 57.71 % of new records were assigned to existing record clusters, which suggests that those author names refer to authors already covered by the existing clusters; 23.94 % were assigned to brand new clusters, which indicates those author names represent new authors; and nearly 18.35 % records belonged to more than one record clusters, which caused cluster merging and helped to fix the recall error.

Finally, we focus on IncAD's ability to discover new authors. Figure 13 shows the (standard) precision, recall and F1 values for new author discovery with the DBLP data over 10 years.

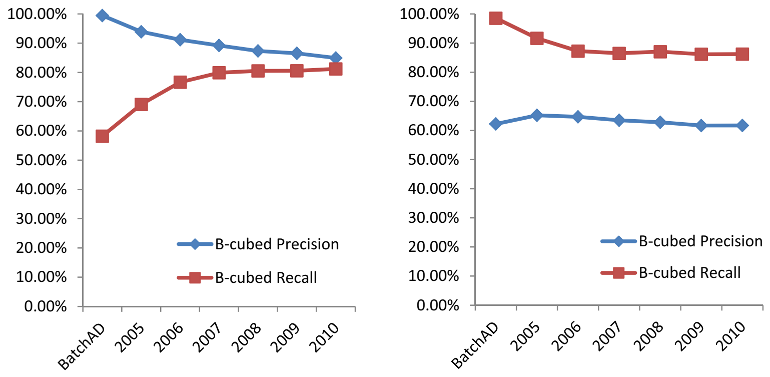

Fig. 12 IncAD with low-recall BatchAD (*left*) and with low-precision BatchAD (*right*) on DBLP (with update period = 7 days)

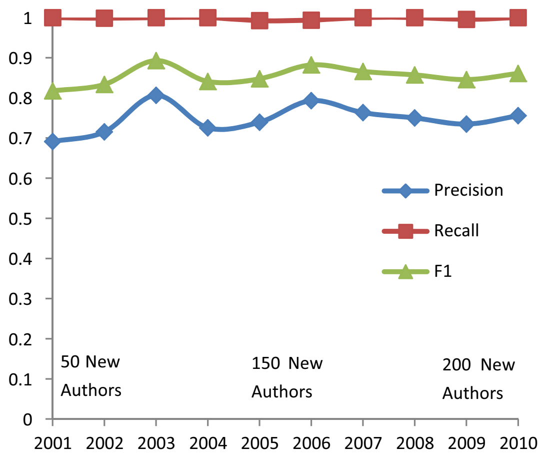

Fig. 13 The accuracy of new author discovery (with update period = 7 days)

In this analysis, precision is the proportion of record clusters that IncAD correctly detected as representing new authors to the total number of record clusters that IncAD identified as new authors. Recall is the proportion of correctly identified record clusters regarding as new authors to the total number of actual new authors. It can be observed that IncAD achieves near 100 % recall and 70–80 % precision. Thus, IncAD almost never misses an emerging researcher, but sometimes fails to recognize authors who have already published papers in the past. According to a close inspection, we found that such errors often occur when an author changes her affiliation and/or starts a new research topic. However, note that our BatchAD+IncAD framework is often able to correct such errors (i.e. merge existing clusters) after it has received more evidence from new updates, as we have demonstrated in Fig. 12.

6 Conclusions

In this paper, we proposed a BatchAD+IncAD approach to author name disambiguation for handling ever-growing digital libraries. Our approach processes existing data in one go and then processes new data periodically. It is highly efficient and practical compared to the BatchAD-only approach, and maintains high accuracy. BatchAD clusters existing records without using any prior knowledge, and subsequently IncAD processes new records in three ways: assign new records to existing record clusters, create new clusters, or merge existing clusters and add the new records to the merged clusters. We used one large unlabeled data set for efficiency evaluation, and two labeled data sets for accuracy evaluation. Our main findings are as follows.

1. In terms of efficiency, the BatchAD+IncAD approach far outperforms the BatchAD-only approach. The major IncAD cost is on the "data loading" and "author model update" stage, so a relatively long update period is more economical. Moreover, the efficiency of our BatchAD significantly outperforms five state-of-the-art batch AD methods, while our IncAD significantly outperforms three incremental AD methods.
2. In terms of accuracy, the BatchAD+IncAD approach is comparable to the Batch-only method (with only a less than 3 % loss), while a relatively short update period tends to

have higher accuracy. IncAD can substantially work well on not only our BatchAD algorithm but also on other batch methods (*k*-means, DBSCAN, DISTINCT, CollectiveER and ProbModel). Moreover, IncAD can improve the accuracy even when the BatchAD results have low recall.

3. According to IncAD's classification of unlabeled data, about 57.71 % of new records belong to existing clusters, 23.94 % belong to new clusters, and the other 18.35 % cause merging of existing clusters. Moreover, IncAD achieves nearly 100 % recall and about 70–80 % precision in the task of discovering new authors with the labeled test data (*DBLP*).

In conclusion, our proposed BatchAD+IncAD can efficiently and accurately disambiguate author names in an ever-growing digital library environment. Since very frequent updates enable high accuracy but incur high costs, an update period such as 7 days or longer appears to be a practical choice according to our experiments. In future work, we would like to explore methods for improving other components: for example, it would be interesting to incorporate an active learning mechanism in IncAD by utilizing a web search engine for resolving uncertain authorships.

Acknowledgments The authors are grateful to the reviewers for reviewing this paper. The authors thank Professor Jim Kirk from the computer science department of Union University for revising the English expressions of the paper. The authors also thank the Microsoft Academic Search team for providing the API for data accessing. This research is supported by the National Science Foundation of China under Grant Nos. 61173112, 60921003, 91118005, 91218301, 61221063; National High Technology Research and Development Program of China under Grant No. 2012AA011003; Key Projects in the National Science and Technology Pillar Program under Grant Nos. 2011BAK08B02, 2011BAK08B05, 2012BAH16F02; National Key Technologies R&D Program of China under Grant No. 2013BAK09B01.

References

Amigó, E., Gonzalo, J., Artiles, J., & Verdejo, F. (2009). A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Information Retrieval*, *12*, 461–486.

Balog, K., Azzopardi, L., & de Rijke, M. (2009). Resolving person names in web people search. *Weaving services and people on the World Wide Web*, *1*, 301.

Bhattacharya, I., & Getoor, L. (2007). Collective entity resolution in relational data. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, *1*(1), 5.

Bollen, J., Rodriguez, M. A., Van De Sompel, H., Balakireva, L. L., & Hagberg, A. A. (2007). The largest scholarly semantic network...ever. In *Proceedings of WWW'07* (pp. 1247–1248).

Byung-won, O., & Lee, D. (2007). Scalable name disambiguation using multi-level graph partition. In *SIAM international conference on data mining*.

Byung-won, O., Elmacioglu, E., Lee, D., Kang, J., & Pei, J. (2006). An effective approach to entity resolution problem using quasi-clique and its application to digital libraries. In *Proceedings of JCDL'06* (pp. 51–52).

Byung-won, O., Lee, D., Kang, J., & Mitra, P. (2005). Comparative study of name disambiguation problem using a scalable blocking-based framework. In *ACM/IEEE joint conference on digital libraries* (pp. 344–353).

Cheng, Y., Chen, Z., Wang, J., Agrawal, A., & Choudhary, A. (2013). Bootstrapping active name disambiguation with crowdsourcing. In *Proceedings of the 22nd ACM international conference on information & knowledge management* (pp. 1213–1216). ACM.

Culotta, A., Kanani, P., Hall, R., Wick, M., & McCallum, A. (2007). Author disambiguation using error-driven machine learning with a ranking loss function. In *Workshop on information integration on the Web - WIIW. ACM.*

de Carvalho, A. P., Ferreira, A. A., Laender, A. H. F., & Gonçalves, M. A. (2011). Incremental unsupervised name disambiguation in cleaned digital libraries. *Journal of Information and Data Management*, *2*(3), 289.

Esperidião, L. V. B., Ferreira, A. A., Laender, A. H. F., Gonçalves, M. A., Gomes, D. M., Tavares, A. I., & de Assis, G. T. (2014). Reducing fragmentation in incremental author name disambiguation. *Journal of Information and Data Management*, *5*(3), 293.

Fan, X., Wang, J., Bing, L., Zhou, L., & Hu, W. (2008). Ghost: An effective graph-based framework for name distinction. In *Proceedings of CIKM'08* (pp. 1449–1450).

Godoi, T. A., Torres, R. S., Carvalho, A. M. B. R., Gonçalves, M. A., Ferreira, A. A., Fan, W., & Fox, E. A. (2013). A relevance feedback approach for the author name disambiguation problem. In *Proceedings of the 13th ACM/IEEE-CS joint conference on digital libraries* (pp. 209–218). ACM.

Gong, J., & Oard, D. W. (2009). Selecting hierarchical clustering cut points for web person-name disambiguation. In *Proceedings of the 32nd international ACM SIGIR conference on research and development in information retrieval* (pp. 778–779). ACM.

Han, H., Zha, H., & Giles, C. L. (2003). A model-based k-means algorithm for name disambiguation. In *International semantic web conference*.

Han, H., Zha, H., & Giles, C. L. (2005). Name disambiguation in author citations using a k-way spectral clustering method. In *ACM/IEEE joint conference on digital libraries* (pp. 334–343).

Han, H., Zha, H., Li, C., Tsioutsiouliklis, K., & Giles, C. L. (2004). Two supervised learning approaches for name disambiguation in author citations. In *ACM/IEEE joint conference on digital libraries* (pp. 296–305).

Huang, J., Ertekin, S., & Giles, C. L. (2006). Efficient name disambiguation for large-scale databases. In *Proceedings of the 10th European conference on principles and practice of knowledge discovery in databases* (pp. 536–544).

Jiang, L., Wang, J., An, N., Wang, S., Zhan, J., & Li, L. (2009). Two birds with one stone: A graph-based framework for disambiguating and tagging people names in web search. In *Proceedings of WWW'09* (pp. 1201–1202).

Jinha, A. E. (2010). Article 50 million: An estimate of the number of scholarly articles in existence. *Learned Publishing*, *23*(3), 258–263.

Joachims, T. (1999). Making large-scale svm learning practical. *Advances in Kernel Methods* (pp. 169–184). Cambridge, MA: MIT Press.

McRae-Spencer, D. M., & Shadbolt, N. R. (2006). Also by the same author: Aktiveauthor, a citation graph approach to name disambiguation. In *Proceedings of the 6th ACM/IEEE-CS joint conference on digital libraries* (pp. 53–54). ACM.

Monz, C., & Weerkamp, W. (2009). A comparison of retrieval-based hierarchical clustering approaches to person name disambiguation. In *Proceedings of SIGIR'09* (pp. 650–651).

Na, S., Lee, S., Jung, H., Kim, P., Sung, W., & Lee, J. (2009). On co-authorship for author disambiguation. *Information Processing and Management*, *45*, 84–97.

Nguyen, H., & Cao, T. (2008). Named entity disambiguation: A hybrid statistical and rule-based incremental approach. *The Semantic Web*, *5367*, 420–433.

Pereira, D. A., Ribeiro-neto, B. A., Ziviani, N., Laender, A. H. F., Gonçalves, M. A., & Ferreira, A. A. (2009). Using web information for author name disambiguation. In *ACM/IEEE joint conference on digital libraries* (pp. 49–58).

Qian, Y., Hu, Y., Cui, J., Zheng, Q., & Nie, Z. (2011). Combining machine learning and human judgment in author disambiguation. In *Proceedings of the 20th ACM international conference on information and knowledge management* (pp. 1241–1246). ACM.

Rahm, E., & Bernstein, P. A. (2001). A survey of approaches to automatic schema matching. *The VLDB Journal*, *10*(4), 334–350.

Song, Y., Huang, J., Councill, I. G., Li, J., & Giles, C. L. (2007). Efficient topic-based unsupervised name disambiguation. In *ACM/IEEE joint conference on digital libraries* (pp. 342–351).

Tan, Y. F., Kan, Min, y., & Lee, D. (2006). Search engine driven author disambiguation. In *ACM/IEEE joint conference on digital libraries* (pp. 314–315).

Tang, J., Alvis Cheuk, M., Fong, B. W., & Zhang, J. (2012). A unified probabilistic framework for name disambiguation in digital library. *Knowledge and Data Engineering, IEEE Transactions on*, *24*(6), 975–987.

Torvik, V. I., & Smalheiser, N. R. (2008). Author name disambiguation in medline. *ACM Transactions on Knowledge Discovery from Data*, *3*(3), 11.

Treeratpituk, Pucktada, & Giles, L. C. (2009). Disambiguating authors in academic publications using random forests. In *ACM/IEEE joint conference on digital libraries* (pp. 39–48).

Treeratpituk, P. (2012). *Person name disambiguation in the multicultural and online setting*. Pennsylvania: The Pennsylvania State University.

Wang, F., Li, J., Tang, J., Zhang, J., & Wang, K. (2008). Name disambiguation using atomic clusters. In *Proceedings of the 9th international conference on web-age information management* (pp. 357–364).

Wang, X., Tang, J., Cheng, H., & Adana, P. S. Y. (2011). Active name disambiguating. In *Proceedings of 2011 IEEE international conference on data mining*.

Yang, K. H., Jiang, J. Y., Lee, H. M., & Ho, J. M. (2006). Extracting citation relationships from web documents for author disambiguation. Technical report, Technical Report (TR-IIS-06-017).

Yin, X., Han, J., & Yu, P. S. (2007). Object distinction: Distinguishing objects with identical names. In *Data engineering, 2007. ICDE 2007. IEEE 23rd international conference on*, (pp 1242–1246). IEEE.

Zhang, R., Shen, D., Kou, Y., & Nie, T. (2010). Author name disambiguation for citations on the deep web. *Web-Age Information Management*, *6185*, 198–209.