Reducing Fragmentation in Incremental Author Name Disambiguation

Luciano Vilas Boas Esperidião[1,2], Anderson A. Ferreira[1], Alberto H. F. Laender[3], Marcos André Gonçalves[3], David Menotti Gomes[1], Andrea Iabrudi Tavares[1], Guilherme Tavares de Assis[1]

[1] Departamento de Computação - Universidade Federal de Ouro Preto, Brazil
[2] Departamento de Computação - Instituto Federal de Minas Gerais, Brazil
luciano.espiridiao@ifmg.edu.br,{ferreira,menotti,andrea.iabrudi,gtassis}@iceb.ufop.br
[3] Departamento de Ciência da Computação - Universidade Federal de Minas Gerais, Brazil
{mgoncalv,laender}@dcc.ufmg.br

Abstract. Author name ambiguity is a hard problem that occurs when several authors publish articles with the same name or when a same author publishes their articles under different names. Traditionally, automatic disambiguation methods process the author names of all citation records in a repository. Aiming efficiency, incremental methods disambiguate author names only when new citation records are inserted into the repository. As a side effect, several citation records of a same author may be associated with different authors, aka, the fragmentation problem. To diminish this problem, we propose a new merge-oriented incremental method capable of reducing such side effect, without the need to apply a traditional disambiguation method on the whole repository. Our experimental evaluation shows that our method produces significant improvements when compared to an incremental baseline and is very competitive with batch-mode methods.

Categories and Subject Descriptors: H.3.3 [**Information Search and Retrieval**]: Information Search and Retrieval; H.3.7 [**Information Storage and Retrieval**]: Digital Libraries

Keywords: author name ambiguity, bibliographic citation, incremental disambiguation

1. INTRODUCTION

Author name ambiguity is a hard problem that occurs when several authors publish articles with the same name (homonyms) or when the same author publishes articles under different names (synonyms). The reasons for name ambiguity include name changes, multiple non-roman name transliterations, typographic errors, lack of standards and common practices, and decentralized content generation.

Name ambiguity deeply affects the quality of scholarly digital libraries (DLs), such as DBLP[1], MEDLINE[2] and BDBComp[3]. These DLs contain millions of bibliographic citation records. Each record represents one publication and has many attributes like authors names, work and publication venue titles, and publication year. Authorship identification aims to assign citation records to authors, based on their names. Due to name ambiguity, automatic authorship identification methods may assign to an author publications of other people, resulting in impure DL repositories. It may also split publications of a same author under slightly different names, thus fragmenting her publications

[1] http://dblp.uni-trier.de
[2] http://www.ncbi.nlm.nih.gov/pubmed
[3] http://www.lbd.dcc.ufmg.br/bdbcomp

This research is partially funded by InWeb - The Brazilian National Institute of Science and Technology for the Web (MCT/CNPq/FAPEMIG grant number 573871/2008-6), and by the authors's individual grants from CAPES, CNPq, and FAPEMIG.

across the DL repository. One direct effect of this problem is wrong citation analysis due to incorrect calculation of citations counts received by the publications of a specific author.

Traditional automatic disambiguators aim to reduce wrong assignments in a repository [Ferreira et al. 2012b] by disambiguating all author names at the same time. More formally, let $C = \{c_1, c_2, ..., c_k\}$ be a collection of *citation records*. Each citation record c_i has a list of *attributes* which includes at least author names, work title, publication venue title and publication year. Each attribute has a specific value composed of a list of *elements*. An element of the attribute "author names" is the name of a single author. Each author name element is a *reference* r_j to an author. The disambiguator partitions the set of author references $\{r_1, r_2, \ldots, r_m\}$ into a set of reference clusters $A = \{a_1, a_2, \ldots, a_n\}$. A reference cluster a_i should contain all and only references to a given author.

More recently, incremental methods [Carvalho et al. 2011] have been proposed to disambiguate author names only for new citation records inserted into a DL repository, being potentially more efficient and practical than traditional methods that disambiguate a whole DL in a single pass [Ferreira et al. 2012b]. On the other hand, an incremental disambiguator assigns a newly inserted reference r_j to either an existing cluster $a_i \in A$ or a new cluster a_k, which is added to A. This latter case happens when the reference belongs to an author with no previous publication in the repository.

An important question regarding incremental methods is: *How can we keep the purity of the clusters while reducing fragmentation?* A cluster is pure if it contains references to a unique author and it is fragmented when author references are split in several clusters. Incremental disambiguation focuses on the purity of the clusters, since an erroneous assignment of a reference to a given cluster may mistakenly assign other references to this same cluster. An undesired side effect of forcing cluster purity is an increase of cluster fragmentation.

Sporadic disambiguation of the full repository may alleviate cluster fragmentation. This solution has a high computational cost as DL repositories may contain millions of records. Moreover, all eventual manual disambiguation performed by a DL administrator may be lost, as they are usually not considered by these methods when disambiguating the whole repository. Alternatively, from time to time, we may check for compatible clusters in the repository in order to merge them. This solution is also very expensive since we must check all clusters to perform the merges.

This work addresses the fragmentation problem that occurs when applying incremental disambiguators. To do so, we propose a new merge-oriented incremental method that uses the newly record inserted into the DL repository to identify potential fragmented clusters and merge them. However, if the clusters to be merged include references to different authors, the resulting cluster will be even more impure. Therefore, we should not use all references in the clusters to estimate cluster similarity as this may increase the similarity of clusters of different authors. Thus, comparing the new record with a small set of representative references from the clusters will avoid the undesired merge of clusters of different authors and keep purity. Accordingly, we also propose different strategies to select few representative references from each cluster aiming to further improve final purity. We evaluate our method using real and synthetic collections, and identify the strategies that, when compared to our baselines, significantly reduce fragmentation while maintaining the purity in the clusters.

In sum, the main contributions of this paper are: (1) a new incremental author name disambiguator that uses a newly inserted record to reduce fragmentation; (2) the proposal of reference selection strategies for improving cluster purity; and (3) an experimental evaluation of our method, using real and synthetic collections, that compares the proposed strategies for selecting representative reference considering several representative state-of-the-art baselines.

The rest of this article is organized as follows. Section 2 discusses related work. Section 3 describes our method and the proposed strategies for selecting representative reference. Section 4 presents our experimental evaluation. Finally, Section 5 presents our conclusions and discusses possible directions for future work.

2. RELATED WORK

In the literature, we distinguish between methods that disambiguate all references in the DL repository and those that handle only the references of newly inserted citation records, called incremental methods. The first group can be further divided in *author grouping* and *author assignment* methods [Ferreira et al. 2012b]. *Author grouping methods* rely on the similarity between reference attributes to guide their grouping into clusters that belong to same author [Bhattacharya and Getoor 2007; Cota et al. 2010; Fan et al. 2011; Han et al. 2005; Huang et al. 2006; Kang et al. 2009; Soler 2007; Song et al. 2007; Torvik and Smalheiser 2009; Treeratpituk and Giles 2009; On et al. 2006]. *Author assignment* methods [Ferreira et al. 2012c; Ferreira et al. 2014; Han et al. 2004; Han et al. 2005; Tang et al. 2012; Veloso et al. 2012] aim at directly assigning the references to their respective authors.

Author grouping methods comprise clustering techniques where the critical component is the similarity function applied to the attributes of two references. They group all references of a same author by maximizing intra and minimizing inter-cluster similarities. The similarity function may be predefined [Bhattacharya and Getoor 2007; Cota et al. 2010; Han et al. 2005; Soler 2007; Wu et al. 2014], learned using a supervised machine learning technique [Huang et al. 2006; Peng et al. 2012; Torvik and Smalheiser 2009; Treeratpituk and Giles 2009; Wang et al. 2011], or extracted from the relationships among authors, usually represented as a graph [Fan et al. 2011; Levin and Heuser 2010; Shin et al. 2014; On et al. 2006].

Author assignment methods directly assign each reference to a given author by building a model that represents the authorship likelihood given the citation attributes (author names, publication venue, terms in the work title). They use either a supervised classification technique [Ferreira et al. 2012c; Ferreira et al. 2014; Han et al. 2004; Veloso et al. 2012] or a model-based clustering technique [Han et al. 2005; Tang et al. 2012].

INDi [Carvalho et al. 2011] is the only known incremental disambiguation method proposed in the literature. Unlike traditional disambiguation methods, it disambiguates references only at insertion time. INDi compares each reference in the newly inserted record with representative clusters already identified in the DL repository to check whether the reference belongs to a known cluster. If the reference is compatible with a pre-existing cluster, it is assigned to such a cluster. Otherwise, the reference is considered to point to a previously unreferenced author in the DL and a new cluster is created for it. INDi applies a few heuristics on coauthor names, work title and publication venue title to decide whether a reference and a cluster are compatible.

Since INDi adopts the strategy of assigning a reference to a single cluster, it may split author references into several clusters, causing fragmentation of the repository. Our work proposes strategies that tackle both incremental disambiguation and the fragmentation problem in an innovative way. Our method incrementally disambiguates the new references inserted into a DL repository while trying, at the same time, to reduce fragmentation. To reach such an objective, our method uses only the new references (citations) inserted into a DL repository as evidence to identify fragmented (split) clusters and merge them, when it judges this is appropriate.

3. INCREMENTAL AUTHOR NAME DISAMBIGUATION

In this section, we present our method for incremental author name disambiguation which aims at reducing fragmentation in DL repositories. First, we describe the basic incremental method that assigns each reference in a new inserted record to an existing reference cluster of a compatible author. Next, we show how our new method improves the basic one by selecting the reference clusters most compatible with the newly inserted record in order reduce fragmentation. Finally, we describe our proposed strategies for selecting representative references to avoid reducing purity.

Algorithm 1 Basic Incremental Disambiguation

Input: Set of reference clusters A; Citation record c;
Output: Set of reference clusters A;
1: $c' \leftarrow$ PreprocessCitationRecord(c);
2: **for each** reference $r \in c'$ **do**
3: $a \leftarrow$ selectCluster(A,r);
4: **if** $a = \emptyset$ **then**
5: $a \leftarrow$ newCluster();
6: $A \leftarrow A \bigcup \{a\}$;
7: **end if**
8: add(a, r);
9: **end for**

3.1 Basic Incremental Author Name Disambiguation

Let $C = \{c_1, c_2, ..., c_k\}$ be a collection of *citation records* in a DL repository, $R = \{r_1, r_2, \ldots, r_m\}$ be a set of references from C and $A = \{a_1, a_2, \ldots, a_n\}$ be a set of reference clusters in the repository, where each cluster is considered as belonging to an author and has a *representative name* obtained from the author name attribute of its references. Algorithm 1 describes the basic incremental author name disambiguation method. When a citation record c is inserted into a DL repository, for each reference r (an author name occurrence) from c, an incremental disambiguator selects a compatible cluster a and inserts r in a. Ideally, a contains references to the same author. If none of the clusters are selected (*i.e.*, none of them is *similar enough* to r), the disambiguator considers it as belonging to a new author, creates a new cluster with r and adds this cluster to A. As already mentioned, this basic method may assign references to a same author to distinct clusters, thus increasing fragmentation and reducing the quality of the whole DL repository.

3.2 Merge-oriented Incremental Author Name Disambiguation

Our work focuses on reducing fragmentation during the insertion of a new record by merging clusters compatible to the references in such a record. The main idea of our method can be summarized as follows: if a reference from a newly inserted record has high probability of belonging to some existing clusters, there is also a high likelihood that these clusters belong to a same author. So these clusters should be merged to decrease the fragmentation of the repository. Algorithm 2 describes our method. This algorithm receives as input a set of reference clusters A from the repository and a newly inserted citation record c. After a preprocessing (line 1), *i.e.*, after removing stop-words and applying stemming in the elements of the work and publication venue titles, the algorithm generates, for each r from c, a list of candidate clusters A' (line 3), *i.e.*, clusters whose representative author name is similar to r's author name. The selection of candidate clusters avoids the comparison of all attributes among dissimilar author names. Next, the algorithm selects a set of clusters S from A' that likely contains references to the same author of r (line 4) using all attributes. If S is empty, the algorithm considers r as belonging to a new author (lines 5-7), otherwise it merges all clusters from S into a (lines 8-14). Finally, it inserts the reference r in a (line 16).

We use the function proposed by Carvalho et al. [2011] for comparing the newly inserted references with existing clusters. This function is described by Algorithm 3. A cluster a is compatible to a reference r if a includes references with common coauthors and similar work or publication venue titles. Algorithm 3 uses α_{Title} and α_{Venue} to check the similarity of titles. If r or a has no coauthors, the function only checks similarity between work titles or publication venue titles. In this case, it increases the similarity thresholds, α_{Title} and α_{Venue}, by a factor δ (lines 5-9) to strength the similarity requirements for inserting r into a. The similarity between author names is evaluated by a function derived from the Fragment Comparison algorithm, an edit-distance matching algorithm specially designed for persons' names [Cota et al. 2010]. For work or publication venue titles, we use the cosine similarity function.

Algorithm 2 Merge-oriented Incremental Disambiguation

Input: Set of reference clusters A; Citation record c;
Output: Set of reference clusters A;

```
1: c' ← preprocessCitationRecord(c);
2: for each reference r ∈ c' do
3:    A' ← getCandidateClusters(A, r);
4:    S ← selectClusters(A',r);
5:    if S=∅ then
6:       a ← newCluster();
7:       A ← A⋃{a};
8:    else
9:       A ← A − S;
10:      a ← ∅;
11:      for each cluster s ∈ S do
12:         a ← a⋃s;
13:      end for
14:      A ← A⋃{a};
15:   end if
16:   add(a, r);
17: end for
```

Algorithm 3 Comparison Function

Input: Cluster a of references; Reference r; Similarity thresholds α_{Venue} and α_{Title}; Incremental value δ;

```
1: if similarAuthorName(a, r) then
2:    if similarCoauthors(a, r) and (similarTitle(a, r, α_Title) or similarVenue(a, r, α_Venue)) then
3:       return TRUE;
4:    else
5:       if r.coauthorList is empty or a.coauthorList is empty then
6:          auxThresVenue ← α_Venue + δ;
7:          auxThresTitle ← α_Title + δ;
8:          if (similarTitle(a, r, auxThresTitle) or similarVenue(a, r, auxThresVenue)) then
9:             return TRUE;
10:         else
11:            return FALSE;
12:         end if
13:      else
14:         return FALSE;
15:      end if
16:   end if
17: else
18:   return FALSE;
19: end if
```

Our goal is to identify fragmented clusters using only the references from the new inserted record. If these clusters are merged, we decrease fragmentation and avoid disambiguating the full repository. As cluster purity is a concern, first our *selectClusters* function selects a cluster a similar to a reference r using the given thresholds to compare work and publication venue titles. To select other clusters, we increase these thresholds by a δ factor. As discussed before, the more references from the clusters we use to compare with the new inserted reference, the higher the chance of merging clusters of different authors, decreasing the repository purity. We envision several strategies to select representative references from each cluster to use in the comparison process. In Subsection 4.4 we describe the strategies used in our experimental evaluation.

3.3 Complexity Analysis

Since the similarity functions are applied on small strings and they are not affected by the size of the collection, we analyse the complexity of our method by estimating the number of comparisons

performed by it.

For each reference r from the new citation record c, our method selects a set of candidate clusters A with cost $O(n_a)$, where n_a is the number of authors (i.e., clusters) in the DL. Next, our method selects from A the most likely clusters to contain references to the same author of r, performing $O(n_s)$ comparisons, where n_s is the number of clusters in A', which is usually much smaller than n_a. To avoid comparisons with each reference in a cluster, all clusters are represented similarly to a citation record, with attributes derived by grouping the values of the attributes of all references within the cluster. Thus, in the comparison process, we compare the attribute values from r with the cluster attributes. In our experimental evaluation, we call the function *selectClusters* to return all compatible clusters or only the two most similar compatible clusters. In the last case, the most expensive one, we compare reference r with each reference from each compatible cluster in A' with cost $O(n_c \times n_p)$, where n_c is the number of compatible clusters and n_p is the maximum number of references in a cluster.

To merge the compatible clusters in S, we add all references from each compatible cluster to a unique cluster with cost $O(n_c \times n_p)$. The selection of representative references from each cluster is performed only when we insert r in a cluster and each strategy has a cost that we will refer as z. Thus, for each reference we have the cost $O(n_a + n_s + n_c \times n_p + n_c \times n_p + z)$. As n_s is very smaller than n_a, the number of compatible clusters n_c is very small and a citation has a few author names (i.e., we may consider the number of author names as a constant), the cost of Algorithm 2 is $O(n_a + n_p + z)$

4. EXPERIMENTAL EVALUATION

In this section, we discuss the results of a set of experiments we conducted to evaluate our method and the proposed strategies for selecting representative references. We first describe the collections, evaluation metrics and baselines used. Then, we discuss the effectiveness of our method in comparison with the baselines.

4.1 Collections

We use both real and synthetic collections to evaluate the strategies for reducing fragmentation in incremental author name disambiguation. Two collections of references were gathered from the BDBComp and DBLP digital libraries. Other ones were generated by SyGAR [Ferreira et al. 2012a], a synthetic data generator that simulates an evolving digital library.

KISTI. This collection[4] was built by the Korean Institute of Science and Technology Information [Kang et al. 2011] for English homonyms author name disambiguation. It comprises the citation records from the top 1000 most frequent author names from late-2007 DBLP. A reference was built for each author name in each citation record. The manual disambiguation relied on Google to retrieval authors' personal publication pages. For each reference, a query composed of the author surname and the work title was submitted to Google. Manual inspection of the first retrieved web pages identified the correct personal publication page. This collection has 37,613 citation records, 881 groups of same-name persons and 6,921 authors.

BDBComp. This collection[5] was created by us based on the Brazilian Digital Library of Computing. It comprises 363 records associated with 184 distinct authors: about 2 records by author. Despite its small size, this collection is difficult to disambiguate, with many authors having only one or two citation records. It contains the 10 largest ambiguous groups in this repositoy considering the period between 1987–2007.

[4]Available at *http://www.kisti.re.kr*.
[5]Available at *http://www.lbd.dcc.ufmg.br/lbd/collections/disambiguation*.

Table I. Distribution of average number of publications per year per author (DBLP: 1984 - 2008).

	Average Number of Publications per Year			
	One	Two	Three	Four
New Authors	55%	30%	10%	5%
Existing Authors	14%	42%	28%	16%

SyGAR. In order to evaluate our method in scenarios in which references to new authors are continously inserted into a DL and new inserted records reflect changes in the authors' profiles, simulating changes in their research interests over time, we used four collections generated by SyGAR[6]. The main input of SyGAR is a collection of previously disambiguated citation records, referred to as the *input collection*. Each record in such a collection comprises the three attributes most commonly exploited by disambiguation methods, namely, a list of author names, a list of unique terms present in the work title, and the publication venue title [Ferreira et al. 2012b]. Authors with the same ambiguous name, and their corresponding records, are organized into *ambiguous groups* (e.g., all authors with name "C. Chen"). There are also configuration parameters, such as number of loads, number of records per load, probability of selecting a *new* coauthor, that must be set. As output, SyGAR produces a representative list of synthetically generated citation records, in which each created record consists of the three aforementioned attributes.

As *input collection* for SyGAR, we used a previous collection of citation records from DBLP [Cota et al. 2010; Ferreira et al. 2014], which sums up 4,272 records associated with 220 distinct authors, corresponding to an average of approximately 20 records per author. The original version of this collection was created by Han et al. [Han et al. 2004], who manually labeled its records. Ten data loads, representing data inserted into the collection in each year, were generated. The number of records generated for each author was based on the distribution presented in Table I, extracted from the DBLP input collection. The first state, before the ten loads, was generated with the same number of references in the input collection.

We want to evaluate two scenarios with the SyGAR collections: (1) addition of references to new authors, *i.e.,* authors without any previous publication in the DL repository, and (2) changes in the authors' publication profile, *i.e.,* changes in the topics in which the authors publish. For the first scenario, we generated two collections, SyntheticNew5 and SyntheticNew10, in which each load added a set of references to new authors corresponding to, respectively, 5% and 10% of the total number of current authors in the DL. For the second scenario, we also generated two collections, SyntheticChange10 and SyntheticChange50, in which, for each load, respectively 10% and 50% of the authors changed their profiles.

4.2 Metrics

The K metric determines the trade-off between the average cluster purity (ACP) and the average author purity (AAP). ACP evaluates the purity of the empirical clusters with respect to the theoretical clusters. Thus, if the empirical clusters are pure, the corresponding ACP value will be 1. ACP is defined as: $\text{ACP} = \frac{1}{N}\sum_{i=1}^{e}\sum_{j=1}^{t}\frac{n_{ij}^2}{n_i}$, where N is the total number of references, t is the number of theoretical clusters, e is the number of empirical clusters, n_i is the total number of references in the empirical cluster i, and n_{ij} is the total number of references in the empirical cluster i which are also in the theoretical cluster j.

AAP evaluates the fragmentation of the empirical clusters with respect to the theoretical clusters. If the empirical clusters are not fragmented, the corresponding AAP value will be 1. AAP is defined as: $\text{AAP} = \frac{1}{N}\sum_{j=1}^{t}\sum_{i=1}^{e}\frac{n_{ij}^2}{n_j}$, where n_j is the number of references in the theoretical cluster j.

[6]Available at *http://www.lbd.dcc.ufmg.br/lbd/collections/sygar.*

Thus, the K metric consists of the geometric mean between ACP and AAP values. It evaluates the purity and cohesion of the empirical clusters extracted by each method, being defined as: $K = \sqrt{\text{ACP} \times \text{AAP}}$.

4.3 Baselines

In this work, we used INDi (Incremental author Name Disambiguation) [Carvalho et al. 2011] as our incremental baseline in addition to two traditional author grouping (batch-mode) representative methods and two state-of-the-art supervised author assignment methods. For the best of our knowledge, INDi is the unique method proposed for incrementally disambiguating author names. It follows the basic incremental author name disambiguation (Subsection 3.1). INDi works in an incremental way prioritizing the purity of the clusters, *i.e.*, in case of doubts INDi assigns the reference to a new author (*i.e.*, new cluster) instead of assigning to a pre-existing author. To assign references, INDi tries to find a cluster with similar author name, at least one similar coauthor name and similar work or publication venue titles. It uses the fragment comparison algorithm [Cota et al. 2010] to measure the similarity among author and coauthor names and cosine distance among work and publication venue titles. The INDi complexity is $O(n_r \times n_a)$ [Carvalho et al. 2011], where n_r is the total number of references in the new load and n_a is the total number of authors (clusters) in the DL repository.

The two traditional author grouping methods are HHC (Heuristic Hierarchical Clustering) [Cota et al. 2010] and LASVM-DBSCAN [Huang et al. 2006]. HHC is a two-step based method. Its first step creates clusters of references with similar author names that share at least a similar coauthor name. This step produces very pure but fragmented clusters. Then, in the second step, it successively merges clusters of references with similar author names according to the similarity between the other citation attributes (*i.e.*, work title and publication venue). In each round of merging, the information of merged clusters is aggregated providing more information for the next round. This process is successively repeated until no more merges are possible according to a similarity threshold. LASVM-DBSCAN uses DBSCAN [Ester et al. 1996] for clustering references by author. First, the distance metric between pairs of citations (similarity vectors) used by DBSCAN is calculated by a trained online active support vector machine algorithm (LASVM). The authors use different functions for each different attribute to make the similarity vectors. In our work, we use cosine for work and publication venue title and soft-TFIDF for author and coauthor names. We also used the LaSVM package [Bordes et al. 2005] and DBSCAN available from Weka[7].

The two traditional author assignment methods are SVM (method based on Support Vector Machines) [Han et al. 2004] and SLAND (Self-training Lazy Associative Name Disambiguator) [Veloso et al. 2012]. SVM associates each author name (individual person) with an author class and train the classifier for that class. Each reference is represented by a feature vector with the elements of their attributes (author and coauthor names, and terms of work and publication venue titles) and their TFIDF (Term Frequency - Inverse Document Frequency) as the feature weight. The authors used the "one class versus all others" approach to multi-class classification. SLAND infers the author of a reference by using a supervised rule-based associative classifier. The method uses author names, work title and publication venue title attributes as features and infers the most probable author of a given reference r_i using the confidence of the association rules $\mathcal{X} \rightarrow a_i$ where $\mathcal{X}$ only contains features of r_i. The method also works on demand, i.e., association rules to infer the correct author of a reference are generated in the moment of disambiguation. The method is also capable of inserting new examples into the training data during the disambiguation process, using reliable predictions, and detecting authors not present in the training data[8].

[7] *http://www.cs.waikato.ac.nz/ml/weka/*

[8] Although this baseline was proposed by our group, it is currently the most effective state-of-the-art method found in the literature for the author name disambiguation problem.

4.4 Strategies for Selecting Representative References

In this subsection, we describe the proposed strategies to select the representative references of each cluster. These strategies discard noisy references, reducing their impact in the merge process while preserving cluster purity. This potentially avoids the effect that wrongly assigned references might have when merging clusters.

We represent each reference as a feature vector and each feature corresponds to an author name or a term from a work or publication venue title. We use TF-IDF to calculate the feature values and measure the distance between references by means of the cosine distance.

The five proposed strategies are:

—Using all references in the cluster (ALL) - This strategy does not discard any reference from the cluster. As we show in the experiments, this technique is useful when the clusters are very pure.
—Using a time window (TIW) - This strategy uses a time window to select the references in each cluster that will take part of the comparison with the references of a new citation record. Our hypothesis is that the most recent references are more likely to better reflect the current interests of the author, also reflected in the newly inserted record. To use this strategy, we need to define the value of a parameter w: the size of the window. Thus, we select references whose publication years are at most w years apart from the new record's publication year.
—Using DBSCAN to filter noise (DBS) - We use the DBSCAN [Ester et al. 1996] to identify noisy references in a cluster and discard them from similarity evaluation of the new reference r. We need to specify two parameters: the minimum number of points and the radius.
—Using K-means to filter noise (KMS) - This strategy applies the K-means clustering technique to each cluster producing subgroups (*i.e.*, clusters of a cluster). We discard the outliers, *i.e.*, the references whose distance to their subgroup centroid exceeds a threshold. To use this strategy we need to specify the number of groups produced by K-means and the threshold distance.
—Using references closest to the centroid (CEN) - This strategy selects a percentage p of the references in each cluster. These selected references are the p% closer to the centroid of the cluster, calculated as the average of the feature vector of all references in the clusters. To use this strategy, we need to specify the parameter p.

To decide whether a cluster and a reference are similar, we use the same function proposed Carvalho et al. [2011] (Algorithm 3). Finally, we evaluate two merging strategies: to merge the two clusters most compatible to the new reference r (TMC) or to merge all compatible clusters (ALL). To select the two most compatible clusters, we calculate the distance from the new reference r to each cluster. The distance between reference r and a cluster is the smallest distance between r and each reference in the cluster.

4.5 Experimental Setup

We performed ten rounds of experiments in each collection for each proposed strategy. In the case of KISTI and BDBComp, each load corresponds to all publications of a given year. In each round, we generated a random permutation of the references in a load, *i.e.*, we changed the order of insertion of each reference in a given year. The disambiguation performance is defined as the average performance over ten rounds with a 99% Student's t-distribution confidence interval. The parameters α_{Title}, α_{Venue} and δ were experimentally tuned for each collection and kept fixed for all strategies. Table II shows these parameter values for each collection.

We adopt the notation <clusters to merge>-<selection strategy> to specify the evaluated strategy. <clusters to merge> corresponds to the number of clusters to merge, *i.e.*, the two most compatible clusters (TMC) or all compatible clusters (ALL). <selection strategy> identifies the strategy applied

Table II. The parameter values used for each collection.

Collection	α_{Title}	α_{Venue}	δ
BDBComp	0.0	0.2	0.2
KISTI	0.0	0.0	0.2
SyntheticNew5	0.1	1.0	0.2
SyntheticNew10	0.1	0.9	0.2
SyntheticChange10	0.1	0.9	0.2
SyntheticChange50	0.0	0.2	0.2

Table III. Results in KISTI. Best results, including statistical ties, are highlighted in bold.

Strategy	# of merges	wrong merges	% of wrong merges	*ACP*	*AAP*	*K*
INDi	0	0	-	**0.980 ± 0.000**	0.384 ± 0.001	0.614 ± 0.001
ALL-ALL	2416	114	4.7	0.976 ± 0.000	**0.542 ± 0.002**	**0.727 ± 0.002**
ALL-DBS	2581	94	3.6	0.978 ± 0.000	0.492 ± 0.002	0.694 ± 0.001
ALL-KMS	2419	109	4.5	0.977 ± 0.000	0.509 ± 0.003	0.705 ± 0.002
ALL-TIW	3193	150	4.7	0.978 ± 0.000	0.509 ± 0.002	0.705 ± 0.001
ALL-CEN	3303	149	4.5	0.977 ± 0.000	0.522 ± 0.002	0.714 ± 0.001
TMC-ALL	2162	106	4.9	0.977 ± 0.000	0.526 ± 0.002	0.717 ± 0.002
TMC-DBS	2185	86	3.9	0.979 ± 0.000	0.473 ± 0.002	0.681 ± 0.001
TMC-KMS	2162	98	4.5	0.978 ± 0.000	0.495 ± 0.002	0.696 ± 0.002
TMC-TIW	2097	85	4.1	**0.980 ± 0.000**	0.454 ± 0.002	0.667 ± 0.001
TMC-CEN	2086	86	4.1	**0.980 ± 0.000**	0.459 ± 0.001	0.670 ± 0.001

to select the representative references from the clusters to compare with the new reference. The selection strategies considered are: ALL (based on all references), DBS (based on DBSCAN), KMS (based on K-means), TIW (based on a time window) and CEN (based on the centroid). The strategies based on DBSCAN (DBS) use $minpts = 2$ (minimum number of points) and $\epsilon = 0.4$ (radius). The strategies based on K-means (KMS) use $k = 6$ and the distance from the centroid equals to 0.4. The strategies based on time window (TIW) use size of the window equals to 3. These values were empirically defined.

4.6 Results

Table III shows the final ACP, AAP, and K results for each evaluated strategy applied on KISTI after the last load in the repository. The total number of merges and incorrect merges, and the percentage of incorrect merges performed by each strategy are also presented. In this collection, we notice that merging all clusters similar to a given reference produces slightly better results than merging the two most similar clusters. Using all references from the potential clusters to evaluate their similarity also produces good results (strategies ALL-ALL and TMC-ALL). In this collection, the purity of the clusters (ACP) produced by INDi and all proposed strategies is similar and high. This motivates the use of all information from the clusters and the merge of all clusters similar to the new reference to obtain the best results. Among the tested strategies, ALL-ALL, *i.e.,* merging all similar clusters and using all information of the clusters, statistically outperforms INDi and all other strategies. Compared with INDi, the gain provided by the ALL-ALL strategy is around 16.6% for the K metric, while the worst strategy (TMC-TIW) outperforms INDi by 8.6%. Notice that, in KISTI the ACP confidence interval is too small (less than 0.0001), and, therefore is shown as 0.000.

Table IV shows the results for BDBComp. INDi and all proposed strategies produce very pure clusters (ACP above 0.993). Thus, using all references in the clusters (strategies ALL-ALL and TMC-ALL) also produces the best results for this collection. Strategies ALL-ALL and TMC-ALL outperform INDi around 1.8% and 1.4%, respectively. The small number of references in the BDBComp collection leads to fewer merges and, consequently, to small gains. In this collection, the strategies based on time window (strategies ALL-TIW and TMC-TIW) performed worst than INDi. In this collection, the strategies based on time window greatly limit the number of references in the clusters to be compared with the new reference.

In the SyntheticNew5 collection, we added references to new authors in a rate of 5% per load. As

Table IV. Results in BDBComp. Best results, including statistical ties, are highlighted in bold.

Strategy	# of merges	wrong merges	% of wrong merges	*ACP*	*AAP*	*K*
INDi	0	0	-	**0.994 ± 0.005**	0.765 ± 0.013	0.872 ± 0.005
ALL-ALL	5	1	20.0	**0.995 ± 0.003**	**0.793 ± 0.003**	**0.888 ± 0.002**
ALL-DBS	6	1	16.7	**0.995 ± 0.003**	0.766 ± 0.004	0.873 ± 0.003
ALL-KMS	4	1	25.0	**0.995 ± 0.003**	**0.790 ± 0.002**	**0.886 ± 0.002**
ALL-TIW	11	1	09.1	**0.995 ± 0.003**	0.723 ± 0.005	0.848 ± 0.003
ALL-CEN	7	1	14.3	**0.993 ± 0.005**	0.766 ± 0.014	0.872 ± 0.006
TMC-ALL	4	1	25.0	**0.995 ± 0.003**	0.785 ± 0.004	0.884 ± 0.002
TMC-DBS	4	1	25.0	**0.995 ± 0.003**	0.759 ± 0.008	0.869 ± 0.005
TMC-KMS	4	1	25.0	**0.995 ± 0.003**	0.788 ± 0.002	0.885 ± 0.002
TMC-TIW	11	1	09.1	**0.995 ± 0.003**	0.723 ± 0.005	0.848 ± 0.003
TMC-CEN	4	1	25.0	**0.993 ± 0.005**	0.756 ± 0.019	0.866 ± 0.009

Table V. Results in SyntheticNew5. Best results, including statistical ties, are highlighted in bold.

Strategy	# of merges	wrong merges	% of wrong merges	*ACP*	*AAP*	*K*
INDi	0	0	-	0.927 ± 0.007	0.741 ± 0.007	0.829 ± 0.006
ALL-ALL	325	135	41.5	0.861 ± 0.017	0.819 ± 0.005	0.840 ± 0.009
ALL-DBS	364	157	43.1	0.858 ± 0.022	0.817 ± 0.002	0.837 ± 0.012
ALL-KMS	215	41	19.1	**0.934 ± 0.005**	0.802 ± 0.007	0.865 ± 0.003
ALL-TIW	897	388	43.3	0.783 ± 0.029	0.800 ± 0.004	0.791 ± 0.014
ALL-CEN	884	414	46.8	0.747 ± 0.034	**0.850 ± 0.002**	0.796 ± 0.018
TMC-ALL	321	131	40.8	0.860 ± 0.017	0.819 ± 0.005	0.839 ± 0.009
TMC-DBS	247	58	23.5	0.921 ± 0.005	0.822 ± 0.007	**0.870 ± 0.002**
TMC-KMS	212	40	18.9	**0.934 ± 0.005**	0.801 ± 0.007	0.865 ± 0.003
TMC-TIW	380	131	34.5	0.883 ± 0.015	0.755 ± 0.005	0.816 ± 0.007
TMC-CEN	228	54	23.7	0.924 ± 0.004	0.814 ± 0.009	**0.868 ± 0.003**

Table VI. Results in SyntheticNew10. Best results, including statistical ties, are highlighted in bold.

Strategy	# of merges	wrong merges	% of wrong merges	*ACP*	*AAP*	*K*
INDi	0	0	-	**0.836 ± 0.009**	0.715 ± 0.008	0.773 ± 0.007
ALL-ALL	466	251	53.9	0.706 ± 0.018	0.814 ± 0.010	0.758 ± 0.014
ALL-DBS	503	264	52.5	0.705 ± 0.015	0.807 ± 0.013	0.754 ± 0.013
ALL-KMS	350	105	30.0	**0.850 ± 0.012**	0.806 ± 0.008	**0.827 ± 0.008**
ALL-TIW	1225	651	53.1	0.568 ± 0.018	0.821 ± 0.004	0.683 ± 0.010
ALL-CEN	1239	701	56.6	0.558 ± 0.018	**0.854 ± 0.003**	0.690 ± 0.011
TMC-ALL	459	248	54.0	0.706 ± 0.016	0.812 ± 0.011	0.757 ± 0.013
TMC-DBS	350	116	33.1	0.826 ± 0.008	0.822 ± 0.006	**0.824 ± 0.006**
TMC-KMS	338	98	29.0	**0.851 ± 0.012**	0.805 ± 0.008	**0.828 ± 0.009**
TMC-TIW	490	225	45.9	0.733 ± 0.008	0.754 ± 0.014	0.743 ± 0.009
TMC-CEN	325	89	27.4	**0.839 ± 0.010**	0.814 ± 0.006	**0.826 ± 0.007**

we can see from Table V, the clusters produced by INDi and all proposed strategies are not as pure as those produced in KISTI or BDBComp, which makes more promising the use of techniques to select the information from the clusters and merging the two most compatible clusters. Thus, except for TMC-ALL and TMC-TIW, merging only the two most compatible clusters keeps the purity of the clusters (ACP) compared with INDi. All strategies reduce fragmentation (AAP) with gains from 14.7% (ALL-CEN) to 7.9% (ALL-TIW). It is worth to notice that the strategies based on K-means (TMC-KMS and ALL-KMS) improve purity while reducing fragmentation. Considering the K metric, TMC-DBS and TMC-CEN outperform INDi and all other strategies, with a gain around 4.9% when compared with INDi.

Table VI shows the disambiguation performance in the SyntheticNew10 collection. Notice that, by increasing the rate of new authors from 5% (SyntheticNew5-Table V) to 10% (SyntheticNew10-Table VI), we reduce the purity of the clusters even further because of the increasing ambiguity. Overall, strategies ALL-KMS, TMC-DBS, TMC- KMS and TMC-CEN outperform INDi (around 7%) and all other strategies under the K metric. The strategies ALL- KMS and TMC-KMS also improve purity around 1.7%.

With SyntheticChange10 and SyntheticChange50 we analyzed the impact of changing the authors' publication profile at each load (year) in 10% and 50%, respectively. In SyntheticChange10 (Table VII), INDi and the strategy TMC-CEN obtained the best results in terms of purity (ACP), with the strategies based on K-means (ALL-KMS and TMC-KMS) and DBSCAN (TMC-DBS) achieving a performance close to the that obtained by INDi and TMC-CEN. Regarding fragmentation (AAP) and the metric K, ALL-CEN outperforms INDi and all other strategies, but its clusters are the most

Table VII. Results in SyntheticChange10. Best results, including statistical ties, are highlighted in bold.

Strategy	# of merges	wrong merges	% of wrong merges	*ACP*	*AAP*	*K*
INDi	0	0	-	**0.977 ± 0.002**	0.608 ± 0.002	0.771 ± 0.002
ALL-ALL	173	29	16.8	0.964 ± 0.001	0.654 ± 0.003	0.794 ± 0.002
ALL-DBS	172	29	16.9	0.964 ± 0.001	0.654 ± 0.002	0.794 ± 0.001
ALL-KMS	152	16	10.5	0.976 ± 0.003	0.631 ± 0.003	0.784 ± 0.003
ALL-TIW	722	141	19.5	0.914 ± 0.006	0.671 ± 0.003	0.783 ± 0.003
ALL-CEN	709	128	18.1	0.904 ± 0.005	**0.716 ± 0.003**	**0.805 ± 0.004**
TMC-ALL	171	29	17.0	0.964 ± 0.001	0.654 ± 0.003	0.794 ± 0.002
TMC-DBS	145	15	10.3	0,976 ± 0,001	0,647 ± 0,003	0,795 ± 0,002
TMC-KMS	145	15	10.3	0.975 ± 0.005	0.628 ± 0.002	0.783 ± 0.003
TMC-TIW	196	37	18.9	0.963 ± 0.003	0.587 ± 0.002	0.752 ± 0.001
TMC-CEN	152	15	09.9	**0.977 ± 0.001**	0.637 ± 0.003	0.789 ± 0.001

Table VIII. Results in SyntheticChange50. Best results, including statistical ties, are highlighted in bold.

Strategy	# of merges	wrong merges	% of wrong merges	*ACP*	*AAP*	*K*
INDi	0	0	-	**0.885 ± 0.002**	0.565 ± 0.002	0.707 ± 0.001
ALL-ALL	439	247	56.3	0.793 ± 0.004	0.619 ± 0.006	0.701 ± 0.003
ALL-DBS	394	197	50.0	0.827 ± 0.007	0.592 ± 0.004	0.699 ± 0.004
ALL-KMS	1087	641	59.0	0.884 ± 0.003	0.540 ± 0.005	0.691 ± 0.004
ALL-TIW	352	151	42.9	0.698 ± 0.007	0.583 ± 0.004	0.638 ± 0.003
ALL-CEN	1162	766	65.9	0.602 ± 0.007	**0.680 ± 0.004**	0.640 ± 0.003
TMC-ALL	424	235	55.4	0.799 ± 0.003	0.618 ± 0.004	0.703 ± 0.002
TMC-DBS	297	130	43.8	0.882 ± 0.005	0.595 ± 0.009	**0.724 ± 0.007**
TMC-KMS	333	145	43.5	**0.890 ± 0.002**	0.539 ± 0.004	0.693 ± 0.003
TMC-TIW	350	157	44.9	0.870 ± 0.002	0.485 ± 0.003	0.649 ± 0.002
TMC-CEN	302	117	38.7	**0.890 ± 0.003**	0.561 ± 0.006	0.707 ± 0.003

impure (ACP). TMC-DBS keeps the purity of the clusters similar to INDi and reduces the fragmentation with a gain around 3% under the K metric. In SyntheticChange50 (Table VIII), only TMC-DBS outperforms INDi under the K metric with a gain around 2.4%. TMC-CEN is statistically tied with INDi under all metrics.

Overall we can conclude that, if we have collections whose clusters are impure, an option to attenuate this problem is to merge only the two clusters most similar to the new reference and select the information in each cluster using the strategies based on K-means, centroid or DBSCAN. ALL-KMS, TMC-KMS and TMC-CEN outperformed INDi and all other strategies under the K metric in all collections, but SyntheticChange10. Under the K metric, TMC-CEN outperformed INDi in all collections, but BDBComp and SyntheticChange10 in which there was a statistical tie.

In Table IX, we compare the TMC-CEN strategy, the one with the best overall performance in our experiments, with two traditional (batch-mode) representative methods, HHC and LASVM-DBSCAN[9]. The traditional methods use all references from the first year to last year altogether, meaning that they use much more information than the incremental methods, but also incurring in much higher costs and in the loss of eventual manual corrections that were made in the repositories. In the BDBComp and KISTI collections, HHC outperforms TMC-CEN and LASVM-DBSCAN under the K metric. However, notice that under ACP metric (purity), our proposed strategy TMC-CEN outperforms all baselines, *i.e.*, the strategy TMC-CEN produces the most pure clusters, with gains up to 90% when it is compared with LASVM-DBSCAN in the KISTI collection. As mentioned before, this is important for future merges of the incremental method as new references are included in the repositories. Notice that once clusters of different authors are wrongly merged (decreased purity), it is very hard to identify and fix this mistake later.

In the SyntheticNew5 and SyntheticNew10 collections, strategy TMC-CEN outperforms all baselines under all metrics. Unlike in the BDBComp and KISTI collections that have, respectively, around 3% and 1% of the references with author name in short format (*i.e.,* author name includes only the initial of the first name followed by the last name), SyntheticNew5 and SyntheticNew10 have, respectively, around 50% and 58% of the references with author name in short format[10]. This higher number of author names in short format leads these collections to have more homonyms, hardening the problem.

[9]Remiding that in case of BDBComp and KISTI, the ALL-ALL strategy produced even better results than TMC-CEN due to the initial purity of the clusters. TMC-DBS is also slightly better than TMC-CEN in SyntheticChange50.
[10]All new author names generated by SyGAR are in short format.

Table IX. Comparison of Strategy TMC-CEN with traditional methods. Best results, including statistical ties, are highlighted in bold.

Methods /	TMC-CEN			HHC			LASVM-DBSCAN		
Collections	*ACP*	*AAP*	*K*	*ACP*	*AAP*	*K*	*ACP*	*AAP*	*K*
BDBComp	**0.993±0.005**	0.756±0.019	0.866±0.009	0.853±0.003	**0.983±0.003**	**0.916±0.003**	0.694±0.000	0.794±0.000	0.743±0.000
KISTI	**0.980±0.000**	0.459±0.001	0.670±0.001	0.955±0.001	**0.770±0.003**	**0.858±0.001**	0.514±0.000	0.569±0.000	0.541±0.000
SyntheticNew5	**0.924±0.004**	**0.814±0.009**	**0.868±0.003**	0.647±0.005	0.756±0.011	0.699±0.006	0.588±0.000	0.089±0.000	0.229±0.000
SyntheticNew10	**0.839±0.010**	**0.814±0.006**	**0.826±0.007**	0.512±0.007	0.739±0.020	0.615±0.011	0.764±0.000	0.054±0.000	0.202±0.000
SyntheticChange10	**0.977±0.001**	0.637±0.003	**0.789±0.001**	0.822±0.012	**0.745±0.006**	**0.783±0.008**	0.520±0.000	0.149±0.000	0.278±0.000
SyntheticChange50	**0.890±0.003**	0.561±0.006	0.707±0.003	0.758±0.011	**0.675±0.006**	**0.715±0.007**	0.565±0.000	0.124±0.000	0.265±0.000

Table X. Comparison of TMP-CEN with supervised author assignment methods.

Methods /	TMP-CEN			SVM			SLAND		
Collections	*ACP*	*AAP*	*K*	*ACP*	*AAP*	*K*	*ACP*	*AAP*	*K*
KISTI	**0.980±0.000**	0.459±0.001	0.670±0.001	0.777 ± 0.003	0.905 ± 0.004	0.839 ± 0.003	0.923 ± 0.002	**0.954±0.002**	**0.938±0.002**
BDBComp	**0.993±0.005**	0.756±0.019	**0.866±0.009**	0.540 ± 0.043	0.900 ± 0.021	0.697 ± 0.035	0.830 ± 0.024	**0.933±0.030**	**0.880±0.024**
SyntheticNew5	**0.924±0.004**	0.814±0.009	**0.868±0.003**	0.511 ±0.089	0.763 ± 0.028	0.624 ± 0.048	0.696 ± 0.027	**0.909 ± 0.012**	0.795±0.016
SyntheticNew10	**0.839±0.010**	0.814±0.006	**0.826±0.007**	0.424 ±0.017	0.778 ± 0.012	0.574 ± 0.012	0.697 ± 0.023	**0.912±0.008**	0.797 ± 0.011
SyntheticChange10	**0.977±0.001**	0.637±0.003	0.789±0.001	0.541 ±0.029	0.693 ± 0.016	0.612 ± 0.022	0.774 ± 0.030	**0.855 ± 0.017**	**0.813 ± 0.023**
SyntheticChange50	**0.890±0.003**	0.561±0.006	**0.707±0.003**	0.407 ±0.012	0.644 ± 0.007	0.512 ± 0.008	0.531 ± 0.044	**0.675 ± 0.027**	0.599 ± 0.036

Table XI. Running time of the TMC-CEN and ALL-CEN strategies and INDi disambiguating each collection.

	Method/Strategy		
Collections	INDi	TMP-CEN	ALL-CEN
KISTI	455.106 ± 23.008	569.477 ± 83.417	661.690 ± 85.896
BDBComp	0.103 ± 0.046	0.170 ± 0.087	0.229 ± 0.097
SyntheticNew5	17.438 ± 1.060	21.840 ± 2.066	33.140 ± 1.946
SyntheticNew10	36.761 ± 1.137	39.794 ± 2.833	49.576 ± 1.453
SyntheticChange10	12.153 ± 0.237	18.928 ± 2.090	22.483 ± 0.548
SyntheticChange50	12.738 ± 0.977	20.533 ± 1.952	30.391 ± 1.266

In this case, our strategy TMC-CEN, that compares the new reference with only the references closer to the centroid of each cluster, produces clusters purer than the ones produced by the traditional methods, that use all references. In the SyntheticChange10 and SyntheticChange50 collections, the results of TMC-CEN and HHC, under K metric, are very close, with TMC-CEN again, producing purer clusters. These collections have around 40% of the references with author names in short format. In sum, in cases in which the ambiguity is higher, our merge-oriented method produced superior effectiveness than most of the tested baselines.

The poor performance of LASVM-DBSCAN is mainly due to the small number of attributes used when compared with the original proposed method described by [Huang et al. 2006]. In that work, several other attributes such as affiliation and e-mail were used. Our collections have only the three most common attributes, *i.e.*, author names, work title and publication venue title and, with only these attributes, the similarity functions learned by the LASVM-DBSCAN are not suitably generalizable. Notice that, as the LASVM-DBSCAN results are the same in each round, their confidence interval is equal to 0.000.

Finally, for comparative purposes only, we include in Table X the effectiveness of two representative supervised methods applied to the same collections. For the KISTI and BDBComp collections, we randomly selected 50% of the records as training data and the other ones as test set. For the synthetic collections, the first state was used as training data and the rest as test set. As before, results correspond to the average effectiveness over ten runs with 99% Student's t-distribution confidence interval. Although a direct comparison is not fair, as we use no training and the supervised methods cannot be applied incrementally as originally proposed (which, by the way, is a huge drawback of this type of method), our results are very competitive.

As we can see, TMP-CEN produces the best results in terms of purity in all cases, and in four out of six cases in terms of the K metric. In case of SyntheticChange10, results in terms of K are also very close to the best method in this collection (SLAND) and very superior to SVM. Finally, for KISTI, although TMP-CEN looses in terms of K for SVM and SLAND, purity is much higher for this method, meaning that there is room for further improvements.

To conclude, Table XI shows the running time (seconds) spent by incremental methods, i.e., INDi

and our method using the TMP-CEN and ALL-CEN strategies in each collection. As always, each running time is defined as the average running time over ten runs with 99% Student's t-distribution confidence interval. The experiments were executed on an Intel core i7-2640M machine with a clock of 2.80GHz and 8 GBytes of RAM memory. We notice that both methods were implemented in Java. We can see that in most cases the runtimes are comparable, with a slight advantage for INDi, as expected. In any case, no method took more than a few seconds or minutes to execute.

5. CONCLUSION

In this work, we propose a new merge-oriented incremental author name disambiguator. In this method, we propose to use a newly inserted publication of an author as evidence to merge fragmented clusters of its previous publications, i.e., we merge publications of an author that were previously considered belonging to different authors. The new citation record inserted into the DL repository is the link to merge fragmented clusters.

In our experimental evaluation, we evaluated several strategies to merge such fragmented clusters while trying to preserve their purity. We want to merge fragmented clusters, but with a reduced chance of a mistaken authorship assignment, since merging clusters belonging to different authors is very hard to identify and fix. In general, we noticed that, if there are few publications wrongly assigned to the authors in the repository, we may merge all clusters similar to the newly inserted record preserving the clusters purity and decreasing fragmentation. The situation is different if ambiguity causes errors in authorship definition. In this case, the strategy that merges only the two most compatible clusters and uses similarity measurements based only on references closer to the cluster centroid is the best one. In our experiments, our merge-oriented method outperformed the only incremental author name disambiguation method known in the literature in basically all cases. When compared to more expensive batch-mode methods that disambiguate a whole digital library at once, our proposed method outperformed one method by large margins in all cases and outperformed the other in three out of six cases (with higher purity in all cases), mainly when the levels of ambiguity were higher. Finally, when compared to state-of-the-art supervised methods for the same problem, our solutions showed competitive results without the incurred costs of supervision and with the aforementioned advantages of incrementally disambiguating a DL repository.

As future work, we intend to check the impact of our proposed representative selection strategies in non-incremental author name disambiguation methods. As the strategy performance is sensitive to author ambiguity, it is worthy to investigate other representative reference selection techniques. We also intend to test our strategies in other collections and envision and evaluate other scenarios beyond insertion of new authors or changes in the authors publication profile.

REFERENCES

Bhattacharya, I. and Getoor, L. Collective Entity Resolution in Relational Data. *ACM Transactions on Knowledge Discovery from Data* 1 (1), 2007.

Bordes, A., Ertekin, S., Weston, J., and Bottou, L. Fast Kernel Classifiers with Online and Active Learning. *Journal of Machine Learning Research* 6 (1): 1579–1619, 2005.

Carvalho, A. P., Ferreira, A. A., Laender, A. H. F., and Gonçalves, M. A. Incremental Unsupervised Name Disambiguation in Cleaned Digital Libraries. *Journal of Information and Data Management* 2 (3): 289–304, 2011.

Cota, R. G., Ferreira, A. A., Gonçalves, M. A., Laender, A. H. F., and Nascimento, C. An Unsupervised Heuristic-Based Hierarchical Method for Name Disambiguation in Bibliographic Citations. *Journal of the American Society for Information Science and Technology* 61 (9): 1853–1870, 2010.

Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*. Portland, Oregon, pp. 226–231, 1996.

Fan, X., Wang, J., Pu, X., Zhou, L., and Lv, B. On Graph-based Name Disambiguation. *ACM Journal of Data and Information Quality* 2 (2): 10:1–10:23, 2011.

Ferreira, A. A., Gonçalves, M. A., Almeida, J. M., Laender, A. H. F., and Veloso, A. A Tool for Generating Synthetic Authorship Records for Evaluating Author Name Disambiguation Methods. *Information Sciences* 206 (1): 42–62, 2012a.

Ferreira, A. A., Gonçalves, M. A., and Laender, A. H. F. A Brief Survey of Automatic Methods for Author Name Disambiguation. *SIGMOD Record* 41 (2): 15–26, 2012b.

Ferreira, A. A., Machado, T. M., and Gonçalves, M. A. Improving Author Name Disambiguation with User Relevance Feedback. *Journal of Information and Data Management* 3 (3): 332–347, 2012c.

Ferreira, A. A., Veloso, A., Gonçalves, M. A., and Laender, A. H. F. Self-training Author Name Disambiguation for Information Scarce Scenarios. *Journal of the Association for Information Science and Technology* 65 (6): 1257–1278, 2014.

Han, H., Giles, C. L., Zha, H., Li, C., and Tsioutsiouliklis, K. Two Supervised Learning Approaches for Name Disambiguation in Author Citations. In *Proceedings of the ACM/IEEE-CS Joint Conference on Digital Libraries*. Tucson, USA, pp. 296–305, 2004.

Han, H., Xu, W., Zha, H., and Giles, C. L. A Hierarchical Naive Bayes Mixture Model for Name Disambiguation in Author Citations. In *Proceedings of the ACM Symposium on Applied Computing*. Santa Fe, USA, pp. 1065–1069, 2005.

Han, H., Zha, H., and Giles, C. L. Name Disambiguation in Author Citations using a K-way Spectral Clustering Method. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries*. Denver, CO, USA, pp. 334–343, 2005.

Huang, J., Ertekin, S., and Giles, C. L. Efficient Name Disambiguation for Large-Scale Databases. In *Proceedings of the European Conference on Principles and Practice of Knowledge Discovery in Databases*. Berlin, Germany, pp. 536–544, 2006.

Kang, I.-S., Kim, P., Lee, S., Jung, H., and You, B.-J. Construction of a Large-scale Test Set for Author Disambiguation. *Information Processing and Management* 47 (3): 452–465, 2011.

Kang, I.-S., Na, S.-H., Lee, S., Jung, H., Kim, P., Sung, W.-K., and Lee, J.-H. On Co-authorship for Author Disambiguation. *Information Processing & Management* 45 (1): 84–97, 2009.

Levin, F. H. and Heuser, C. A. Evaluating the Use of Social Networks in Author Name Disambiguation in Digital Libraries. *Journal of Information and Data Management* 1 (2): 183–197, 2010.

On, B.-W., Elmacioglu, E., Lee, D., Kang, J., and Pei, J. Improving Grouped-entity Resolution using Quasi-Cliques. In *Proceedings of the IEEE International Conference on Data Mining*. Hong Kong, China, pp. 1008–015, 2006.

Peng, H.-T., Lu, C.-Y., Hsu, W., and Ho, J.-M. Disambiguating Authors in Citations on the Web and Authorship Correlations. *Expert Systems with Applications* 39 (12): 10521 – 10532, 2012.

Shin, D., Kim, T., Choi, J., and Kim, J. Author Name Disambiguation using a Graph Model with Node Splitting and Merging based on Bibliographic Information. *Scientometrics* 100 (1): 15–50, 2014.

Soler, J. M. Separating the Articles of Authors with the same Name. *Scientometrics* 72 (2): 281–290, 2007.

Song, Y., Huang, J., Councill, I. G., Li, J., and Giles, C. L. Efficient Topic-based Unsupervised Name Disambiguation. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries*. Vancouver, BC, Canada, pp. 342–351, 2007.

Tang, J., Fong, A. C. M., Wang, B., and Zhang, J. A Unified Probabilistic Framework for Name Disambiguation in Digital Library. *IEEE Transactions on Knowledge and Data Engineering* 24 (6): 975–987, 2012.

Torvik, V. I. and Smalheiser, N. R. Author Name Disambiguation in MEDLINE. *ACM Transactions on Knowledge Discovery from Data* 3 (3): 1–29, 2009.

Treeratpituk, P. and Giles, C. L. Disambiguating Authors in Academic Publications using Random Forests. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries*. Austin, TX, USA, pp. 39–48, 2009.

Veloso, A., Ferreira, A. A., Gonçalves, M. A., Laender, A. H., and Meira Jr., W. Cost-effective On-demand Associative Author Name Disambiguation. *Information Processing & Management* 48 (4): 680 – 697, 2012.

Wang, X., Tang, J., Cheng, H., and Yu, P. ADANA: Active Name Disambiguation. In *Proceedings of the International Conference on Data Mining*. Vancouver,Canada, pp. 794–803, 2011.

Wu, H., Li, B., Pei, Y., and He, J. Unsupervised Author Disambiguation using Dempster–Shafer Theory. *Scientometrics*, 2014.