Online Person Name Disambiguation with Constraints

Madian Khabsa
Computer Science and Engineering
The Pennsylvania State Univeristy
University Park, PA, USA
madian@psu.edu

Pucktada Treeratpituk*
Science Park Promotion Agency
Ministry of Science and Technology
Bangkok, Thailand
pucktada@gmail.com

C. Lee Giles
Information Sciences and Technology
Computer Science and Engineering
The Pennsylvania State Univeristy
University Park, PA, USA
giles@ist.psu.edu

ABSTRACT

While many clustering techniques have been successfully applied to the person name disambiguation problem, most do not address two main practical issues: allowing constraints to be added to the clustering process, and allowing the data to be added incrementally without clustering the entire database. Constraints can be particularly useful especially in a system such as a digital library, where users are allowed to make corrections to the disambiguated result. For example, a user correction on a disambiguation result specifying that a record does not belong to an author could be kept as a cannot-link constraint to be used in any future disambiguation (such as when new documents are added). Besides such user corrections, constraints also allow background heuristics to be encoded into the disambiguation process. We propose a constraint-based clustering algorithm for person name disambiguation, based on DBSCAN combined with a pairwise distance based on random forests. We further propose an extension to the density-based clustering algorithm (DBSCAN) to handle online clustering so that the disambiguation process can be done iteratively as new data points are added.

Our algorithm utilizes similarity features based on both metadata information and citation similarity. We implement two types of clustering constraints to demonstrate the concept. Experiments on the CiteSeer data show that our model can achieve 0.95 pairwise F1 and 0.79 cluster F1. The presence of constraints also consistently improves the disambiguation result across different combinations of features.

Categories and Subject Descriptors

H.3.3 [[**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Algorithms, Experimentation

Keywords

Name Entity Recognition; Record Linking; Name Disambiguation; Clustering; Online Disambiguation

*Work done while at PSU

JCDL'15, June 21–25, 2015, Knoxville, Tennessee, USA.

http://dx.doi.org/10.1145/2756406.2756915.

1. INTRODUCTION

People search is a large part of the search traffic on the internet. An old analysis of query logs from AllTheWeb and AltaVista search sites show that 11-17% of the queries were composed of a person name with additional terms and 4% were simply just person names [22, 3]. The most popular query on Google has often been a celebrity name, with other four or five names ranked in the top ten queries. Similar patterns can also be found in digital library search. Around 9% of the search requests made to the digital library CiteSeerX are queries with author names. This number goes up to almost 19% when counting search from unique IPs.

The task of searching the web for information regarding an individual person can be very challenging, especially when the name is shared by many others. According to the data from 1990 U.S. Census Bureau, 90,000 different names are shared by 100 million people [4]. As the amount of information on the web grows, more of these people are mentioned on different web pages, causing even more ambiguity in the search results. This problem is further compounded by the fact that sometimes the same person is referred to differently at various places. A person might be referred to with the full name on his/her homepage, but with just an initialized name or even just their position in some news articles. Moreover, even the information associated with one single person can be different in different sources. For instance, two web pages about the same person may provide different name spellings and different addresses.

Generally, the ambiguity of person's name comes in three varieties: (1) the aliasing problem - when a person uses multiple name variations such as "Ronald W. Williams" and "R.W. Williams", and (2) the common name problem - when there is more than one person with the same name, which is especially problematic for high frequency names such as many asian names; and (3) the typographic error problem - which often results from human input or automatic extraction systems. The goal of person name disambiguation is to

resolve such ambiguities, linking and merging all the records of the same person together.

Name ambiguity is an important problem in many applications, including web search, natural language processing, information integration, and digital libraries. Name disambiguation is a specialized case of a more general database problem, called record linkage. Without name disambiguation, uses in the people search task, are often required to manually sort through search results to find the right person. In digital libraries, it hinders the accurate attribution of scholarly work, which is often used by academic institutions and funding agencies for promotion and funding consideration. In databases, name ambiguity leads to difficulty in merging multiple personal information databases, such as patient medical records from various healthcare providers. Additionally, the resulting disambiguated names can also be used to help improve other data mining results such as natural language processing, and social network analysis. While the person name disambiguation problem has long been well-studied, it has recently gained even more significance due to the increased ubiquity of names on the internet and the rise of social networking sites.

In this work we address two main practical issues found in a working name disambiguation system but generally have not been addressed by most previous person name disambiguation algorithms. First, most previous methods treat the person name disambiguation problem as a static problem, where all entities to be disambiguated are available to the algorithms. However, for working systems such as PubMed, the number of names to be disambiguated is ever increasing, for instance, when new papers are added to a digital library. Most previous methods would require the disambiguation process be periodically run on the whole data, which is time consuming and usually not scalable. Second, often the disambiguated clusters contain inconsistencies either due to insufficient information or mistakes by the similarity function. Thus, we propose an extension to the density-based clustering algorithm (DBSCAN) to handle online clustering so that the disambiguation process can be done iteratively as new data points are added. Additionally, our clustering method also imposes constraints on the clustering result, ensuring that each record in the same disambiguated cluster is consistent with each other. Our experiments show that our method not only outperforms the previous method in terms of performance, but also is capable of discovering new name clusters as new records are added.

The remainder of this paper is organized as follows. Section 2 surveys related work, while Section 3 describes the pairwise profile similarity function. In Section 4 we introduce DBSCAN with constraints and extend it to run in online fashion. The results and evaluations are reported in Section 5. Section 6 concludes the paper and discusses future work.

2. RELATED WORK

Previous work in person name disambiguation can be generally be categorized as either supervised or unsupervised approaches. Unsupervised methods harness some pre-defined likeness between people records in grouping them into clusters (e.g. similarity between topic-distribution or link-structure) [5, 21]. Supervised methods, on the other hand, explicitly learn the linking functions or rules from labeled examples [15, 16, 23, 24]. Some hybrid approaches also have been proposed. [23] and [12] use heuristics to automatically generate reference sets and use them, instead of labeled examples, as training data for the supervised methods. In [20] disambiguation is conducted using heuristics, with supervision being applied to optimize the heuristics parameters. We can also classify person name disambiguation methods based on the types of information they employ. In the digital library domain, the most often-used information is found in citations such as titles, coauthors, and venues [15, 16]. Additional information that is extracted from documents themselves, such as affiliations [16, 24] and abstract [21], have also been used. Some utilize information in the structure of the citation graph and the co-authorship graph [5]. Some explore external knowledge sources such as search engine and author homepages for disambiguation [18].

However the aspect of clustering streaming data has been ignored in most previous work on person name disambiguation. Most, if not all, assume the number of entities to be disambiguated static, and carry out the disambiguation process in a batch mode. On the other hand, many studies have been done on how to cluster streaming data in the database and data mining communities [29, 14, 13, 9, 8, 2, 1]. Ester et al. introduced an incremental version of DBSCAN [10]. While Cao et al. modify an density-based clustering method for dealing with streaming data [7]. Wagstaff and Cardie propose two types of instance-level clustering constraints, must-link and cannot-link, and show that these constraints can increase clustering accuracy and decrease runtime [27]. Whang et al. propose a general framework for entity resolution that can handle some integrity constraints, called "negative rules" [28]. Carlos Ruiz and colleagues introduced C-DBSCAN to accommodate for constraints when clustering data[19], but their work uses instance level constraints only such as must-link and cannot-link. Our work, on the other hand, introduces cluster level constraints in addition to instance level constraints. Furthermore, our work combines a streaming DBSCAN method along with constraints requirements that are not only at the instance level, but also at the cluster level.

3. PAIRWISE PROFILE SIMILARITY

Given two author profiles, p_1 and p_2 we seek to build similarity function that measures if these two profiles refer to the same author or not. In our previous work [24] we have introduced a machine learning method to learn such similarity function using random forests. Random forest is an ensemble classifier proposed by Breiman that combines a collection of decision trees [6]. Each decision tree within the forest is built with a different bootstrap sample drawn from the original data set. Each tree is then constructed to the maximum size without any pruning. The variable selection for each split in the tree is conducted on a randomly selected subset of features, instead of on the full feature set as is usually done in the traditional decision tree. Once the forest is built, the classification can be done by simply aggregating the votes of all trees. We will adapt this work to measure profile similarity with small modifications described later.

The pairwise author disambiguation service provides methods for calculating distance between a pair of author records and for retrieving the ε-neighborhood of an author record. The ε-neighborhood of a record r is a set of records with distance from r less than ε. The pairwise disambiguation

service is composed of two components: the record indexes and the pairwise distance function. The indexes together with heuristic blocking function provide efficient retrieval of candidate records for ε-neighborhood generation. Currently, we simply block and index each record according to the last names and the first initials. While a more sophisticated blocking function should improve both the performance and the efficiency of the algorithm, it is beyond the scope of this work.

We extract the total of 31 features for computing the pairwise similarity. These 31 features can be grouped into six categories: name-related information (names and emails), affiliation, coauthors (names and their affiliations), venue information (venues and years), content (abstracts and titles), keyphrases and citations. Table 1 shows all the metadata used in the pairwise similarity, except citation information. Most metadata, such as author/coauthor names, affiliations, and etc, are extracted from each paper itself. These are available in all author records, albeit sometimes erroneous. The venue information (publication venue and year), on the other hand, is obtained through citation matching, and thus is less populated. Out of 3355 records of the test data set (described later), only half ($\sim$1500) contain venue names and about two-third contains years of publication. The citation information is computed from the global citation graph and thus requires access to the whole CiteSeerX database. Lastly, for the keyphrases, we use the SEERLAB algorithm [26] to extract keyphrases from each paper. The examples of keyphrases extracted by SEERLAB system are shown in Table 1.

The pairwise distance function is learned using a random forest. Since a random forest is a collection of randomized decision trees [6], the percentage of trees voting with the record pairs non-coreferent can be used as the distance measure. We use a similar feature set as proposed in [24], which includes similarities between author names, affiliations, emails, coauthors, papers titles, journals, and publication years. We also use two additional metadata types that were not in the original feature set in [24]: citations and keyphrases. For citations, we compute two features based on citation similarity: #bibliographic coupling and #co-citations. #bibliographic coupling between two author records is the number of papers that are cited by both papers, while #co-citations is the number of times that both papers appear together in the reference sections.

As for the keyphrases, we use the normalized Point-wise Mutual Information (nPMI) to measure the similarity between two keyphrases. The normalized Point-wise Mutual Information (nPMI) for two keyphrases, x and y, is defined as:

$$nPMI(x,y) = \frac{log\frac{p(x,y)}{p(x)p(y)}}{-log\, p(x,y)}$$

The $nPMI$ ranges from -1 to 1. $nPMI(x,y)$ is maximal at 1, when x and y fully co-occur and is minimal at -1 when x and y never co-occur ($p(x,y) = 0$). $nPMI(x,y)$ is 0 if x and y are independent. For a record pair r_1 and r_2, with keyphrases $\{kp_{1i}\}_{i=1..n}$ and $\{kp_{2i}\}_{i=1..m}$ respectively, we compute three nPMI similarity measures: total nPMI similarity, maximum nPMI similarity and average nPMI similarity. For example, the total keyword similarity between r_1 and r_2 is

$$\sum_{i=1..n}\sum_{j=1..m} nPMI(kp_{1i}, kp_{2j})$$

Table 1: Example of metadata used for the disambiguation

Title:
Tuning Memory Performance in Sequential and Parallel Programs

Abstract:
Recent architecture and technology trends have led to a significant and increasing gap between processor and main memory speeds. Caches hide these latencies to some extent, but when cache misses...

Author: Anoop Gupta
Affiliation:
Department of Electrical Eng. Computer Systems Laboratory, Princeton University Stanford University

Coauthors:
- Margaret Martonosi, Dept. of Electrical Eng...
- Thomas E Anderson, Computer Science Division, University of California

Venue: IEEE Computer, **Year:** 1995

Keyphrases:
parallel program, performance monitoring system, cache miss, performance information, program memory, memory bottleneck, program execution time, program data, technology trend

4. CLUSTERING PROFILE: ONLINE DBSCAN WITH CONSTRAINTS

To cluster author profile, we use a density-based clustering algorithm, DBSCAN [11]. DBSCAN defines a cluster as a connected region where data records are *dense*. A region is *dense*, if the number of records within ε distance from its center point (seed record) exceeds a minimum threshold, *minPts*. First, DBSCAN selects a record p that has not yet been assigned a cluster. Then, the ε-neighborhood of p is retrieved. If the ε-neighborhood is *dense*, a new cluster is created for p, otherwise p is marked as noise. If p is part of a cluster, then so is every record in p's ε-neighborhood. Thus, every record in the ε-neighborhood is added to the cluster, so are their *dense* ε-neighborhoods. With this process, the cluster is recursively expanded until it is fully discovered then a new unassigned record is selected to find a new cluster. DBSCAN does not require the number of clusters to be specified beforehand. Thus it is suitable for the author disambiguation task since different ambiguous names contain different numbers of true author clusters. This property also allows us to detect new author clusters, as new records are added. DBSCAN component relies on the pairwise similarity measure (as described in Section 3) in retrieving the ε-neighborhood of any given record p. We use $\varepsilon = 0.35$ and $minPts = 3$ in our implementation.

However, the standard DBSCAN assumes that all data points are already available at the start of the clustering process. This is not the case in the typical digital libraries setting, where new records are continuously added. In addition, the original DBSCAN does not address how to incorporate constraints in the clustering process. Thus in this section, we propose a modification to the standard DBSCAN clustering algorithm that enforce constraints on cluster-membership. We will also propose a merge subroutine to DBSCAN, that

Table 2: Examples of author records that could be mistakenly clustered without cluster-level constraints. A and B belong to the same author, while C is not.

Name
A) Execution Based Evaluation of Multistage Interconnection Networks for Cache-Coherent Multiprocessors **Name:** Akhilesh Kumar **Affil:** Intel Corporation Department of Computer Science, 2200 Mission College Blvd Texas AM University, Santa Clara College Station
B) FFT Implementations on nCUBE Multiprocessor **Name:** A Kumar **Affil:** Department of Computer Science, Texas AM University
C) Real-Time Communication in FDDI-Based Reconfigurable Networks **Name:** Amit Kumar **Affil:** Department of Computer Science, Texas AM University

allows new records to be added to the existing clustering result.

4.1 Types of Constraints and Motivation

In a real author disambiguation system, it generally is desirable to guarantee certain integrity property of each cluster. Table 2. shows an example of records that could be mistakenly clustered together by DBSCAN without an integrity check. In this case, record A and B belong to the same author, while C is of a different author. A and B are very similar on all three metadata (name, affiliation and both are multiprocessor-related). The similarity between B and C is less than the similarity between A and B; their topics are less related. So if

$$distance(A, B) < distance(B, C) < \varepsilon \ll distance(A, C)$$

Since both A and C are within ε-neighborhood of B, they will both be put in the same cluster with B, even though A and C are clearly incompatible and the $distance(A, C) \gg \varepsilon$.

Constraint enforcement can be done in two levels: at the instance-level and at the cluster-level.

Instance-level constraints are rules that check the compatibility between a record pair. With instance-level constraints, only records that are compatible with the seed record are allowed to be included in its ε-neighborhood. It restricts the local connections between a record and its neighbors.

Cluster-level constraints are rules that check the compatibility between a record and a cluster. Cluster-level constraints ensure that every record in a cluster is compatible with each other.

Instance-level constraints are relatively inexpensive and are easy to incorporate. However, it does not guarantee the integrity of the resulting cluster. For example, instance-level constraints would not address the situation shown in Table 2. Cluster-level constraints require more computation to enforce, but guarantee cluster integrity.

We now present two types of constraints currently implemented in the algorithm: the "temporal proximity" heuristic, which is a disjunctive constraint, and the "name compatibility" constraint, which is a conjunctive constraint.

4.1.1 Temporal Proximity

The intuition is that neighbors of a record r should be not only spatially close to r, but also temporally close as well. This is because two author records 10 years apart are unlikely to be coreferent if there are no records in between that time linking them. And also even if they both belong to the same author, their attributes such as coauthors and topics can be quite different because people move in and out of fields and often change collaborators. At the instance-level, temporal proximity requires that a record r can be in the ε-neighborhood of a record p only if r and p are close temporally. At the cluster-level, temporal proximity requires that a record r is a member of a cluster C, only if there exists $p \in C$, such that p is temporally close to r. Temporal proximity is a disjunctive constraint; To satisfy a cluster-level constraint of C, a record only needs to satisfy the instant-level constraint with any records in C. In our implementation, a record pair is considered to be temporally close if their publication dates are within 3 years of each other. Records missing year information are exempted from this constraint.

Table 3: Example of name variations found in CiteSeerX

Name	Found Variations	Note
Chienyu Chen	C Chen, Cy Chen, Chien Yu Chen	Chinese name Segmentation
Chun Che Fung	Lance Chun Che Fung	Extra nickname
David Johnson	Dav ID Johnson	Parsing error
James E Smith	Jim Smith, J E Smith	Nickname
Juan E. Tapiador	Juan M Estevez-Tapiador	Extra last name, confused as middle name
Jocelyn Smith	Jocelyin Smith	Misspelling

4.1.2 Name Compatibility

At the instance-level, name compatibility requires that a record r is a the ε-neighbor of a record p only if the name in r and p are compatible. At the cluster-level, it means that the name of every record in a cluster C must be compatible with each other. Unlike temporal proximity, name compatibility is a conjunctive constraint. To satisfy a cluster-level constraint of C, a record needs to satisfy the instant-level constraint with every record in C.

Table 3 shows examples of names and their variations found in CiteSeerX database. The different variations can be the results of parsing errors and misspelling like in the case of "Dav ID Johnson" and "Jocelyin Smith." Nicknames can be used in place of the first names ("Jim" instead of "James") or in addition to the full name (e.g. "Lance Chun Che Fung"). They can also be cultural dependent. Spanish names often have extra last names, which sometimes are taken as middle names (eg. "Juan M Estevez-Tapiador"), while Chinese names can be segmented and initialized in various ways. These names are all considered compatible with their variations.

Surface name matching is a very challenging problem in itself. Much research is still being devoted to the problem. Here we use the ethnicity-sensitive name matching that we introduced in [25] for enforcing the name compatibility constraint.

4.2 Clustering with Constraints

We now present our modification to the standard DBSCAN to incorporate clustering constraints, called DBSCAN$_C$. DBSCAN$_C$ maintains a data structure that keeps track of constraints for each cluster. When a record is added to

a cluster, it appropriately updates the constraints for that cluster. A cluster-level constraint can be replaced if a strictly more restrictive one is added. For instance, a full middle name constraint would replace a middle name initial constraint. This allows for more efficient constraint checking. An example of constraint pairs that are compatible but one does not subsume the other are first name constraints, "James" and "Jim."

The main procedure of DBSCAN$_C$ is shown in Procedure 1, where D denotes the static collection of records to be disambiguated. The subroutine $query(D, p, \varepsilon)$ retrieves ε-neighborhood for the record p. DBSCAN$_C$ contains two main differences from the standard DBSCAN. First, each ε-neighborhood is sorted in ascending order according to the distance to the seed record p. In the standard DBSCAN, the order that records are processed is irrelevant to the final output. However, with cluster-level constraints, when a record is added to a cluster, it could introduce new constraints that would prevent some records to be added to that cluster later. Since the final result is sensitive to the order of records processed, we heuristically choose to favor the records that are closer to the seed record.

Procedure 1 DBSCAN$_C$(D)

Input: D - static collections of records to be disambiguated

```
 1: mark all records in D as UNVISITED
 2: for all record p in D do
 3:     if p is UNVISITED then
 4:         mark p as VISITED
 5:         N ← query(D, p, ε)
 6:         sort records in N by their distance from p
 7:         N ← IConsFilter(p, N)
 8:         N ← orderedIConsFilter(N)
 9:         if |N| < minPts then
10:             assign p → NOISE
11:         else
12:             expandCluster(p, N)
13:         end if
14:     end if
15: end for
```

Second, each ε-neighborhood is first checked against instance-level constraints, before passing the density test. The call $IConsFilter(p, N)$ checks each record in the N and filters out those that do not satisfy instance-level constraints with the record p. The call $orderedIConsFilter(N)$ removes any record r in N that is not compatible with all the records that precede it in N. In other words, for every record pair $r, s \in N$, if r is closer to p than s, then s must be compatible with r. The reason behind $orderedIConsFilter(N)$ will become more clear when we describe $expandCluster(p, N)$. If the ε-neighborhood of p is sufficiently dense after both integrity checks, the function $expandCluster(p, N)$ is called to create and expand the new cluster.

$expandCluster(p, N)$, shown in Procedure 2, exhaustively expands a new cluster with a starting record p. As in the procedure $DBSCAN_C(D)$, each ε-neighborhood N' is sorted and passed through $IConsFilter$ and $orderedIConsFilter$ filters. An addition $CConsFilter$ filter is also applied to it. $CConsFilter(cid, N')$ imposes cluster-level constraints of the cluster cid to each record in N'. With $IConsFilter$, $orderedIConsFilter$, and $CConsFilter$, we can prove that $\forall$ record $q \in Q$ at all time, q can be added to the cluster cid, without violating any constraints, and that such a property is maintained throughout the clustering process. This property ensures the integrity of each cluster.

Procedure 2 expandCluster(p, N)

```
 1: cid ← nextClusterId()
 2: assign p → cid
 3: Q ← N /* put records in region into a queue */
 4: while Q ≠ ∅ do
 5:     q ← pop a record from Q
 6:     if q is UNVISITED then
 7:         mark q as VISITED
 8:         N' ← query(D, q, ε)
 9:         sort records in N' by their distance from q
10:         N' ← IConsFilter(q, N')
11:         N' ← orderedIConsFilter (N')
12:         N' ← CConsFilter(cid, N')
13:         if |N'| ≥ minPts then
14:             /* append N' to the end of Q */
15:             Q ← Q + N'
16:         end if
17:     end if
18:     if q doesn't belong to any cluster then
19:         assign q → cid
20:     end if
21: end while
```

4.3 Online Disambiguation with Streaming Data

In the online setting, new records can be introduced to the existing clustering result. We add a *mergeRecord* procedure to the algorithm (Procedure 3). The *mergeRecord* procedure allows new records to be added to an existing cluster. The new record could also cause two or more existing clusters to merge together, if it creates a *dense* connection between those clusters. It could also cause a region of records that previously are marked as noises to become sufficiently *dense* to form a new cluster.

mergeRecord first considers the density of the ε-neighborhood of the new record. If the neighborhood is sparse, the new record is marked as noise. Otherwise, if the neighborhood contains existing clusters, the new record is added to the cluster that contains the most records in the ε-neighborhood. If more than one clusters intersect with the ε-neighborhood, they are sequentially merged according to the size of their intersections. After the new record is added to the cluster, and all appropriat clusters are merged, noise records in he neighborhood are added.

5. EVALUATION

5.1 Data & Evaluation Metrics

The CiteSeer's author disambiguation dataset is used for both training and evaluation purposes. The dataset contains author records of 10 highly ambiguous names sampled from the CiteSeer database. The names, their number of records, and their number of unique authors are shown in Table 4. This dataset and its slight variations have been used in many previous works in author disambiguation [16, 21]. The greyed out rows are used in training the random forest model (the same three names were also used in [16] as the training set). The classification accuracy is evaluated

Table 4: CiteSeer author disambiguation collection

	Data	#Rec	#Cluster
1	A. Gupta	498	45
2	A. Kumar	139	31
3	C. Chen	525	99
4	D. Johnson	345	40
5	J. Anderson	307	40
6	J. Robinson	111	27
7	J. Smith	729	83
8	K. Tanaka	52	19
9	M. Jones	348	51
10	M. Miller	226	35

on non-training data, while the clustering performance is evaluated on the whole data set.

We evaluate our disambiguation algorithm using three sets of standard metrics: the pairwise F1 (pF1), the cluster-level F1 (cF1) and the purity/inverse purity measures. The **pairwise F1** is defined as the harmonic mean of the **pairwise precision (pP)** and the **pairwise recall (pR)**. The pairwise precision is the percentage of record pairs placed in the same cluster that are coreferent, while the pairwise recall is the percentage of coreferent record pairs that are discovered by the algorithm (they are placed in the same cluster). Instead of counting pairs, the **cluster-level F1** counts clusters that exactly match the ground truth and is defined as the harmonic mean of the **cluster-level precision (cP)** and the **cluster-level recall (cR)**. The cluster-level precision is the fraction of generated clusters that exactly match those in the ground truth. The cluster-level recall is the fraction of the clusters in the ground truth that are exactly discovered. Lastly, the **purity** measures the purity of each of the generated clusters, while the **inverse purity** measures the fragmentation of clusters in the ground truth in the result. The formal formula of purity is

$$\sum_i \frac{|C_i|}{N} max_j Precision(C_i, L_j)$$

where C_i denotes a cluster in the result, and L_j denotes a cluster in the answer set. Similarly, the inverse purity is defined as

$$\sum_i \frac{|L_i|}{N} max_j Precision(L_i, C_j)$$

We also calculate **ratio of cluster size** *(RCS)*, which is defined as the ratio of the number of clusters retrieved over the number of true clusters.

5.2 Feature Analysis of the Clustering Results

Table 5 shows the comparison of the clustering results using each similarity feature. Two mixture models, MIX and MIX+CKP, are also evaluated. The MIX+CKP model utilizes all features, while the MIX model uses every features except keyphrases and citations. The features used in MIX are similar to those used in other previous work in author disambiguation [16, 24]. All feature sets yield over 89% pairwise-classification accuracies. As expected, the mixture models, MIX and MIX+CKP, yield the highest classification accuracies at 97%. However, other simpler feature sets also give comparable classification accuracies, with name-related features and coauthors features achieving around 95% classification accuracies.

For the clustering performance, the simplistic models do not perform well. The highest pF1 is that of the name model at only 0.65 and the highest cF1 is that of the affiliation model at just 0.54. The coauthors model produces the purest clusters (with Purity at 0.97 and pP at 0.98) but the clusters are quite fragmented (InvPurity at 0.58). This is because a highly similar coauthors connection is a good indicator of coreference, but the lack of such shared connection does not necessary mean that they do not refer to the same person. The venue model also gives a similar clustering result as the coauthors one, relatively pure clusters but very fragmented (high value of InvPurity and RCS). The content-based features such as abstract and keyphrases gave the opposite clustering results. They tend to produce less fragmented clusters but also more impure. Their InvPurity are reasonably high (0.82 and 0.78 respectively), and their Purity are relatively low.

For the mixture models, the MIX model achieves 0.86 pF1 and 0.69 cF1 while the MIX+CKP achieves 0.90 pF1 (+4%) and 0.76 cF1 (+7%), which is significantly higher. The performance of the MIX model is comparable to those of the previously reported result in [16]. The MIX+CKP performs noticeably better in cluster precision and recall (cP and cR).

It is interesting to note that while the classification accuracies of the simplistic models are comparable to those of the mixture models, their clustering performances are significantly lower. The pF1 and cF1 of the name model is just 0.65 and 0.46 respectively, compared to 0.90 and 0.76 for the MIX+CKP model. This difference is even more glaring between the two mixture models. We hypothesize that in the clustering process, misclassification errors will aggregate, thus a small difference in classification accuracy could result in a noticeable difference in the clustering result.

In random forest, one way to measure the importance of a feature in a model is by calculating the average drops in Gini index at nodes where that feature is used as the splitting criteria [6]. Table 7 shows 10 most indicative features in the MIX+CKP model according to this measurement. Middle name is the most informative feature, followed by various affiliation similarities, then keyphrase PMIs and coauthors' affiliations. We hypothesize that the high contributions of these keyphrase PMI features in the MIX+CKP model is what makes the MIX+CKP model better clustering performance over the MIX model. It is interesting to note that the number of papers cited by both records (bibliographic coupling) seems to be more indicative of linkage between authors than the number of co-citations.

5.3 Evaluating Online Disambiguation with Constraints

To evaluate the iterative disambiguation algorithm, we first randomly select 20% of the author records as the initial static records. We then run the disambiguation algorithm over these initial records, producing the initial disambiguated author clusters. The left-over 80% records are then added to the system one-by-one, simulating new incoming data. Table 6 shows the result of the iterative disambiguation algorithm with various constraint settings. Figure 1. shows the graphs of RCS, pF1, cF1, purity and inverse purity of the best configuration (MIX+CKP model with cluster-level constraints) as new records are added.

Table 5: The disambiguation result using different combination of feature sets

Similarity Model	Accuracy	RCS	pP	pR	pF1	cP	cR	cF1	Purity	InvPurity
name	94.6%	2.08	0.69	0.68	0.65	0.28	0.46	0.34	0.83	0.68
affiliation	91.3%	2.47	0.61	0.68	0.54	0.53	0.24	0.54	0.73	0.63
coauthors	93.6%	2.16	0.98	0.48	0.62	0.30	0.61	0.40	0.97	0.58
venue	89.6%	4.43	0.64	0.17	0.25	0.12	0.49	0.19	0.78	0.28
abstract	91.6%	1.07	0.45	0.86	0.52	0.41	0.43	0.40	0.61	0.82
keyphrases	92.5%	1.24	0.46	0.76	0.50	0.36	0.44	0.49	0.65	0.78
citations	92.5%	1.81	0.73	0.63	0.63	0.32	0.57	0.41	0.83	0.67
MIX	96.8%	1.03	0.81	0.94	0.86	0.69	0.69	0.69	0.89	0.87
MIX+CKP	96.9%	1.02	0.85	0.96	0.90	0.76	0.76	0.76	0.92	0.88

Table 6: The disambiguation result of the full-feature similarity and whether heuristic constraints are enforced. LASVM is the online SVM model used in Huang et al[16].

Similarity Model	Constraint	RCS	pP	pR	pF1	cP	cR	cF1	Purity	InvPurity
MIX	none	1.03	0.81	0.94	0.86	0.69	0.69	0.69	0.89	0.87
	instance	1.06	0.85	0.92	0.88	0.69	0.73	0.71	0.91	0.87
	cluster	1.08	0.89	0.94	0.91	0.70	0.74	0.72	0.93	0.87
MIX+CKP	none	1.02	0.85	0.96	0.90	0.76	0.76	0.76	0.92	0.88
	instance	1.06	0.87	0.96	0.90	0.76	0.80	0.78	0.93	0.88
	cluster	1.07	0.95	0.96	0.95	0.76	0.81	0.79	0.97	0.88
LASVM	none	0.94	0.87	0.94	0.91	-	-	0.64	-	-

Procedure 3 mergeRecord(p)

Input: p is a new record added to D, not yet processed

$N \leftarrow query(D, p, \varepsilon)$
sort records in N by their distance from p
$N \leftarrow IConsFilter(p, N)$
if $|N| < minPts$ **then**
  assign $p \rightarrow$ NOISE
else
  $C \leftarrow$ set of clusters C_i, such that $\forall C_i, C_i \cap N \neq \emptyset$
  if $C \neq \emptyset$ **then**
    $L \leftarrow \emptyset$
    for all $C_i \in C$ **do**
      if $\emptyset \neq CConsFilter(i, \{p\})$ **then**
        $L \leftarrow L \cup \{C_i\}$
      end if
    end for
    sort $C_i \in L$ by $|C_i \cap N|$ in descending order
    $C_k \leftarrow$ the cluster $\in L$ with the biggest intersection
  else
    $k \leftarrow nextClusterId()$
  end if
  assign $p \rightarrow k$
  for all C_i in $L \setminus \{C_k\}$ **do**
    if $C_i = CConsFilter(k, C_i)$ **then**
      $C_k \leftarrow C_k \cup C_i$ /* merge C_i to C_k */
    end if
  end for
  $noises \leftarrow \{q | q \in N$ and $q \notin C_i, \forall C_i \in C\}$
  /* $noises$ retained the sorted order of N */
  $noises \leftarrow orderedIConsFilter\ (noises)$
  $noises \leftarrow CConsFilter(cid_0, noises)$
  for all q in $noises$ **do**
    assign $q \rightarrow k$
  end for
end if

Table 7: The top 10 features (out of 31) in the MIX+CKP model according to the average Gini decrease

Rank	Features
1	middle name
2	affiliation (soft-tfidf)
3	affiliation (jaccard)
4	keyword PMI (summation)
5	coauthors' affiliations (jaccard)
6	keyword PMI (max)
7	first name (boolean rule-match)
8	first name (jaro-winkler)
9	affiliation (tfidf)
10	# of bibliographic coupling

Table 6. shows that constraints consistently improve performance across all evaluation matrices. Instance level constraints offer improvement over no constraints configuration, and the cluster level constraints offer improvement over instance level constraints. Cluster level constraints offer 5% increase in pairwise F1 and 3% increase in cluster F1 over the base model. The improvements are most noticeable in pP and cR, and purity.

MIX+CKP model with cluster level constraints achieves 0.95 pairwise F1 and 0.79 cluster F1. Its purity and inverse purity are 0.97 and 0.88. Its pairwise precision is 8% more than those with just instance level constraints, and 10% more than the no constraints setting. 0.81 cluster recall indicates that it successfully recovered 81% of the true clusters, and with 0.76 cluster precision, more than three fourths of its clusters are correct. With 0.97 purity, it rarely mixes multiple people together. While both instance constraints and cluster constraints improve purity, they, however, do not increase inverse purity. RCS is also slightly bigger in constraints conditions. So while constraints help

Table 8: Cluster-level F1 with margins of errors for the MIX+CKP with cluster-level constraints

Margin	cP*	cR*	cF1*
0	0.76	0.81	0.79
1	0.81	0.88	0.84
2	0.83	0.89	0.86

enforce cluster integrity, they do not alleviate the cluster fragmentation problem.

Compared with the previously proposed method in Huang et al [16], LASVM – online SVM, the basic MIX model performs worse in the pairwise F1, 0.86 compared to 0.91 for LASVM, but has higher cluster-level F1, 0.69 compared to 0.64 for LASVM. The basic MIX+CKP performs slightly lower in the pairwise F1, 0.91 compared to 0.91 for LASVM, but performs significantly higher in cluster-level F1, with over 0.12 improvement. With the cluster-level constraints enforced, both the MIX and the MIX+CKP perform better than LASVM model in all evaluation matrices, including both the pairwise F1 and the cluster-level F1.

As new records are introduced to the algorithm, we see the inverse purity goes up in all data sets (Figure. 1). This is because as new information becomes available, the algorithm can better join previously fragmented clusters together. The increases are most noticeable in "cchen," which is the most ambiguous case, and "ktanaka," which is one of the smaller cases. The pairwise F1 and the purity remain relatively stable as records are added. There is more fluctuation in the cluster F1. This is because when new records of a previously unseen author are introduced, it is difficult for a density-based clustering method to pick up on the new cluster. And as more records of that cluster are added, the chance that the algorithm will correctly identify that cluster increases. As new records are added, the number of the true clusters increases. But since RCS remains relatively stable over time, this means that the algorithm is able to identify previously unseen clusters.

In order to better understand the resulting clusters, we modify the definition of the cluster-level measures to include a margin factor. The standard cluster-level measures (cP, cR and cF1) only count the exact match as correct. In the modified measures (denoted cP*, cR* and cF1* respectively), near-exact clusters would also be counted as a match. More precisely, for a given margin M, a non-singleton resulting cluster C and an answer cluster L is considered a match if both $|C - L|$ and $|L - C|$ are less than M. For $M = 0$, it is equivalent to the standard definition. For $M = 1$, a cluster is still considered a match if it is off by only one record. We do not apply the margin in comparing singleton and doubleton clusters (still need a perfect match to count). Table 8 shows the modified cluster-level precision, recall and F1 for the MIX+CKP model with constraints. With the margin $M = 2$, the cluster F1 is over 0.86 with the cluster recall 0.89. This means that almost 90% of author clusters are almost perfectly identified.

Overall, precision and purity are higher than recall and inverse purity. This characteristic is actually desirable in the production system because it is generally easier to merge two (or more) pure clusters and then to split impure clusters, both from the users' and the administrator's perspective. Furthermore, two pure but fragmented clusters could be algorithmically merged if a new author record, whose its ε-neighborhood contains records from both clusters and has above-threshold density, is ingested. But if a cluster is already impure, the standard DBSCAN does not have a mechanism to re-split it.

6. CONCLUSION

Constraints can be particularly useful in a digital library or other situations where users are allowed to make corrections to disambiguated names, especially in an iterative scenario.

We describe an author disambiguation framework that incorporates both metadata information and citation information. We show that our feature set yields high pairwise disambiguation accuracy and could be used successfully in clustering author records. We also propose a novel variation of the DBSCAN-based clustering algorithm that allows external knowledge and constraints to be injected into the disambiguation processes. Two examples of constraints were implemented, one instance-level and one cluster-level constraint, namely the "temporal proximity" and "middle-name compatability." Our experiments demonstrate that our clustering with constraints approach can achieve almost 96% precision and 93% F1 measure in disambiguation accuracy. Additionally, we also propose an extension to our clustering framework, that allows an iterative disambiguation process. Our experiment shows that the iterative disambiguation process is effective. As new people profiles are added, it can use the new information to improve the existing clustering result or to discover new people clusters. Recently we have used a batch variation of this algorithm to disambiguate more than 70 million author mentions in *Medline* [17], and the proposed iterative algorithm can be used to disambiguate newly added records to *Medline*. For future work, one could explore additional types of constraints, both instance-level and cluster-level, and their effect on the disambiguation performance. It would also be interesting to study the effect on efficiency based on how the constraints are enforced.

7. ACKNOWLEDGMENTS

We gratefully acknowledge partial funding from National Science Foundation and useful comments from the referees.

8. REFERENCES

[1] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu. *A framework for clustering evolving data streams*. The 29th VLDB Conference, Sept. 2003.

[2] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu. *A framework for projected clustering of high dimensional data streams*. The 30th VLDB Conference, Aug. 2004.

[3] J. Artiles, J. Gonzalo, and S. Sekine. WePS 2 Evaluation Campaign: Overview of the Web People Search Clustering Task. *2nd Web People Search Evaluation Workshop (WePS 2009), 18th WWW Conference*, Jan. 2009.

[4] J. Artiles, J. Gonzalo, and F. Verdejo. A Testbed for People Searching Strategies in the WWW. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, page 569, New York, New York, USA, 2005. ACM Press.

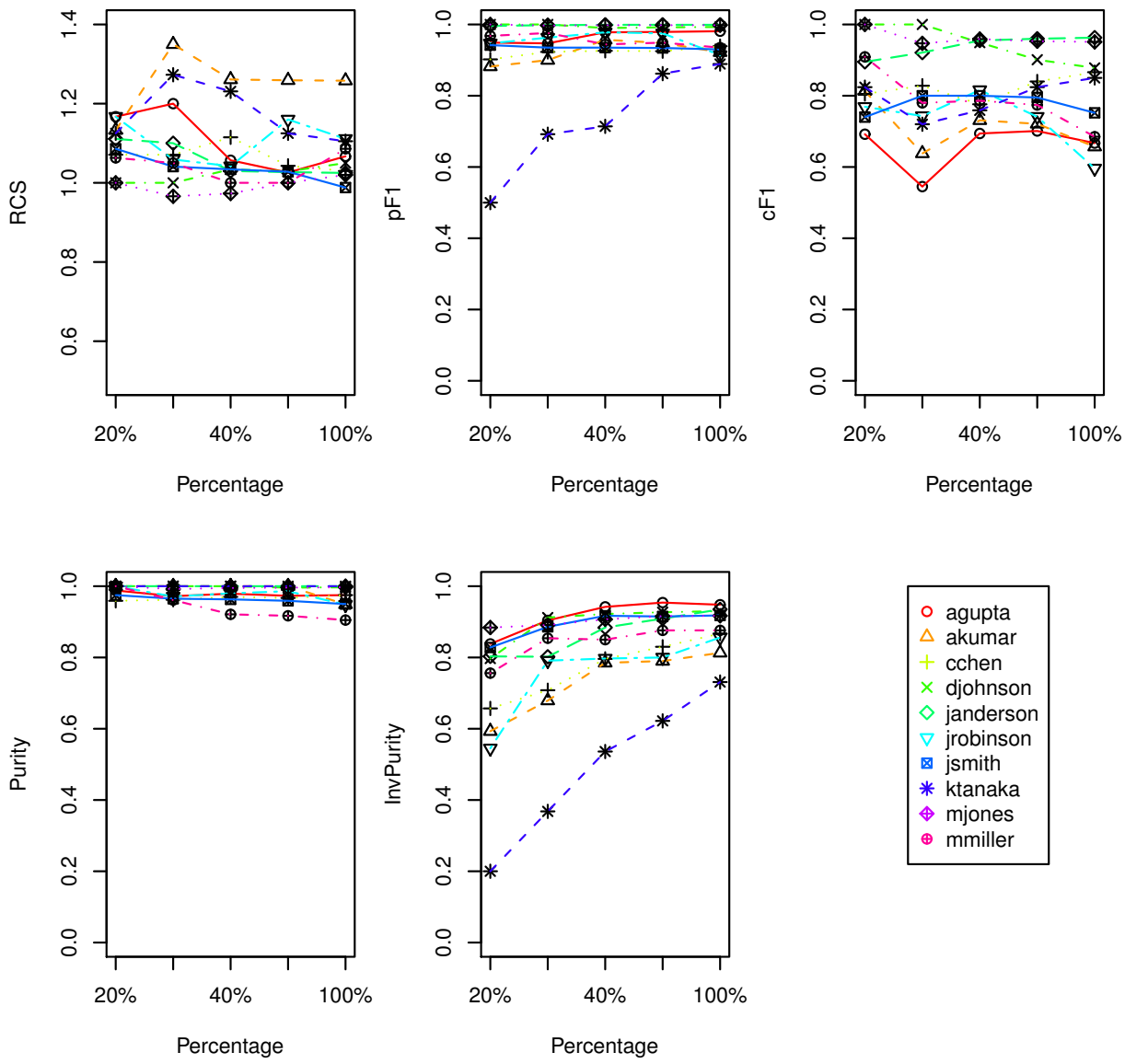

Figure 1: RCS, pF1, cF1, Purity and InvPurity of the MIX+CKP model with cluster-level constraints as new records are added.

[5] R. Bekkerman and A. McCallum. Disambiguating Web appearances of people in a social network. *In Proceedings of The 14th International Conference on World Wide Web (WWW'05)*, pages 463–470, 2005.

[6] L. Breiman. Random Forests. *Machine Learning*, 2001.

[7] F. Cao, M. Ester, W. Qian, and A. Zhou. Density-Based Clustering over an Evolving Data Stream with Noise. *the 6th SIAM International Conference on Data Mining*, May 2006.

[8] M. Charikar, L. O'Callaghan, and R. Panigrahy. Better streaming algorithms for clustering problems. *The 35th annual ACM Symposium on Theory of computing (STOC)*, May 2003.

[9] B.-R. Dai, J.-W. Huang, M.-Y. Yeh, and M.-S. Chen. Clustering on Demand for Multiple Data Streams. *The 4th IEEE International Conference on Data Mining (ICDM'04)*.

[10] M. Ester, H.-P. Kriegel, J. Sander, M. Wimmer, and X. Xu. Incremental clustering for mining in a data warehousing environment. In *VLDB*, volume 98, pages 323–333, 1998.

[11] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[12] A. Ferreira, A. Veloso, and M. Gonçalves. Effective self-training author name disambiguation in scholarly digital libraries. *In Proceedings of the ACM/IEEE Joint Conference on Digital Libraries (JCDL'10)*, 2010.

[13] S. Guha, A. Meyerson, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering data streams: theory and practice. *IEEE Transactions on knowledge and data engineering*, 15(3):515–528, May 2003.

[14] S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering Data Streams. In *The 41st Annual Symposium on Foundations of Computer Science*, pages 359–366. IEEE Comput. Soc.

[15] H. Han, C. L. Giles, H. Zha, C. Li, and K. Tsioutsiouliklis. Two supervised learning approaches for name disambiguation in author citations. *In Proceedings of the ACM/IEEE Joint Conference on Digital Libraries (JCDL'04)*, 2004.

[16] J. Huang, S. Ertekin, and C. L. Giles. Efficient Name Disambiguation for Large-Scale Databases. In *The 10th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2006)*, pages 536–544, 2006.

[17] M. Khabsa, P. Treeratpituk, and C. L. Giles. Large scale author name disambiguation in digital libraries. In *Big Data (Big Data), 2014 IEEE International Conference on*, pages 41–42. IEEE, 2014.

[18] D. Pereira, B. Ribeiro-Neto, and N. Ziviani. Using web information for author name disambiguation. *In Proceedings of the ACM/IEEE Joint Conference on Digital Libraries (JCDL'09)*, 2009.

[19] C. Ruiz, M. Spiliopoulou, and E. Menasalvas. C-dbscan: Density-based clustering with constraints. In *Rough Sets, Fuzzy Sets, Data Mining and Granular Computing*, pages 216–223. Springer, 2007.

[20] A. F. Santana, M. A. Goncalves, A. H. Laender, and A. Ferreira. Combining domain-specific heuristics for author name disambiguation. In *Digital Libraries (JCDL), 2014 IEEE/ACM Joint Conference on*, pages 173–182. IEEE, 2014.

[21] Y. Song, J. Huang, I. G. Councill, J. Li, and C. L. Giles. Efficient topic-based unsupervised name disambiguation. *In Proceedings of the Joint Conference on Digital Libraries (JCDL'07)*, pages 342–351, 2007.

[22] A. Spink, B. J. Jansen, and J. Pedersen. Searching for people on Web search engines. *Journal of Documentation*, 60(3):266–278, 2004.

[23] V. Torvik, M. Weeber, D. Swanson, and N. Smalheiser. A probabilistic similarity metric for Medline records: A model for author name disambiguation. *Journal of the American Society for Information Science and Technology*, 2005.

[24] P. Treeratpituk and C. L. Giles. Disambiguating Authors in Academic Publications using Random Forests. *In Proceedings of the Joint Conference on Digital Libraries (JCDL'09)*, Jan. 2009.

[25] P. Treeratpituk and C. L. Giles. Name-ethnicity classification and ethnicity-sensitive name matching. In *AAAI Conference on Artificial Intelligence*, 2012.

[26] P. Treeratpituk, P. Teregowda, J. Huang, and C. L. Giles. Seerlab: A system for extracting key phrases from scholarly documents. In *Proceedings of the 5th international workshop on semantic evaluation*, pages 182–185. Association for Computational Linguistics, 2010.

[27] K. Wagstaff and C. Cardie. Clustering with instance-level constraints. *Proceedings of the national conference on artificial intelligence*, Apr. 2000.

[28] S. E. Whang, O. Benjelloun, and H. Gracia-Molina. Generic entity resolution with negative rules. *The VLDB Journal*, 18.

[29] J. Yang. Dynamic Clustering of Evolving Streams with a Single Pass. *The 19th International Conference on Data Engineering (ICDE'03)*, Apr. 2003.