The Pennsylvania State University
The Graduate School
College of Engineering

AN API FOR AUTHOR NAME DISAMBIGUATION

A Thesis in
Computer Science and Engineering
by
Gauravi Dudhbhate

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

December 2017

The thesis of Gauravi Dudhbhate was reviewed and approved* by the following:

C. Lee Giles
Professor of Computer Science and Engineering
Thesis Co-Advisor

Kamesh Madduri
Professor of Computer Science and Engineering
Thesis Co-Advisor

Mahmut Kandemir
Professor of Computer Science and Engineering
Graduate Program Chair

*Signatures are on file in the Graduate School.

Abstract

In digital libraries, there are ambiguities present in an author's name primarily when one name can have multiple variations, when multiple authors can share the same name and when the ambiguity exists due to incorrect input of data or due to incorrect extraction by automated software. Especially, in digital libraries, when this problem for author name ambiguity is persistent, it can be inconvenient for users. Authors would then be required to manually sort through the serach result for a scholarly document or an article written by a particular author, in the absence of author name disambiguation techniques.

With great amount of research underway for author name disambiguation, where techniques are achieving almost 90-95 percent accuracy in displaying the accurate articles written by a particular author, the querying latency and the return of results, of such algorithms is slow. Further although such algorithms exist there are very few, if any, end-to-end services that provide a web accessible platform to submit a query and obtain the disambiguated articles with simply the click of a button.

In this thesis, we propose a hierarchical approach to attain a faster querying latency while maintaining the sameaccuracy of 90-95 percent. We further propose an end-to-end service that provides an API to achieve ease in use of the algorithm and user satisfaction. We show that our hierarchical method outperforms the most accurate random forest approach. Finally, we also provide a comparitive analysis of the query latency and the classification accuracy of the two methods.

Table of Contents

List of Figures

List of Tables

Acknowledgments

First, I would like to express my deepest gratitude to my thesis committee, Dr. C. Lee Giles, Dr. Kamesh Madduri for their valuable suggestions, time and effort in guiding me through this thesis. I would also like to specially thanks my advisor, Dr. C. Lee Giles, for his continuing guidance, advice and for giving me an opportunity to work with him.

This thesis would have never come into being without the continual support, patience and time and the intellectual contribution of Kunho Kim. Lastly, I would like to thank my parents and my sister, for their unconditional support throughout my academic career.

Chapter 1
Introduction

1.1 Author Name Disambiguation

Name Disambiguation refers to a entity resolution problem which involves scholarly documents, where the aim is to find all documents belonging to the same entity by clustering them together. In this thesis, we refer entity to an author and the name disambiguation problem is applied to authors in the Pubmed digital library which is described in Chapter 2. The ambiguity in a person's name can be classified into three categories: (1) when a person uses multiple name variations such as Jane-Jane Chen and J J Chen or Jane Chen and J Chen, (2) when there is more than one person with the same name and (3) errors that occur due to human input or automatic extraction software [19]. Similarly ambiguity in digital libraries is exists due to different representations and variations of authors' names, or when multiple authors could share the same name, and when this affects funding considerations it becomes important to resolve this ambiguity. Thus author name disambiguation is therefore, a very popular and interesting problem and various techniques have been applied with an aim to resolve this problem.

A large part of the search traffic on the Internet constitues of people search. Spink et al. [30] answered the question whether personal names formed a major part of the queries to Web search engines. To answer this question the authors analyzed queries submitted by users to two search engines, AllTheWeb and AltaVista. They concluded that around 50 percent of the queries included names of people. Furthermore the most popular query on the Google Search Engine consists of the name of a celebrity.

According to the data from 1990 U.S. Census Bureau, 90,000 different names are shared by 100 million people [1]. Now in the 21st century, this number is ever increasing and is infact difficult to quantify. Therefore, searching for a particular person over the Internet and getting the accurate and appropriate results is quite challenging, especially when the name is shared by many others. With the emergence of Big Data, 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, which constitues the factor of variations in a person′ name. for example, a person could 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 and different information altogether due to a lack of update.

Name ambiguity is an important problem in many applications, including web search, natural language processing, information integration, and digital libraries. Without name disambiguation, searching for people over the web becomes a tedious task, as one has 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 [33].

1.2 API for Author Name Disambiguation

Application Programming Interfaces are advantageous in exposing some of a program′s internal functions to the outside world in a limited fashion. This makes it possible for applications to share data and take actions on one another′s behalf without requiring developers to share all of their software′s code. An API provides a single point of extraction that can be integrated into documents and scientific workflows [22], to allow for easier processing of data. Providing an API to the disambiguation provides easy access for the user, who just has to enter the URL into a browser, thus providing an end-to-end service. Furthermore, a single point of operation with a standard interface allows for improvements in the disambiguation query algorithm to be used by all without the need to rewrite code in order for it to be compatible with new changes.

As scholarly data increases, the number of services to manage and provide access to this data increases. The prevalence of this data had generated the emergence of sites such as Google Scholar [1], CiteSeerX [2], ArXiv [3] and so on. ArXiv, allow users to submit scholarly documents and provide metadata while others, such as CiteSeerX, collect documents by crawling the Web and perform automatic extraction of data. These serve primarily serve a search engines for scholarly document that accept a user query and present data relevant to the query after the search process. The the searching process these sites make use of the metadata of the scholarly documents.

The increasing abundance of scholarly publications, results in problem of data duplication. The reasons of data duplication in scholarly documents and articles can be attributes to the above 3 reasons of name variations, when multiple people have the same name and when errors occur due to incorrect input or incorrect automatic extraction of data as described in Section 1.1. Thus

[1] `www.scholar.google.com`
[2] `http://citeseerx.ist.psu.edu/`
[3] `https://arxiv.org/`

a lot of data can be generated which points to the same author. Thus methods for reducing this data duplication and disambiguating the results a search engine displays are useful since they can improve performance by improving both the accuracy and precision of the search engine.

In this thesis an API for author name disambiguation is presented with an aim to provide a single point of operation with a standard interface for displaying disambiguated results of a user's query. The user's query could consist of the title of a document or the name of an author, in a more general term, it could consist of the data present in a citation of a scholarly document, or the user could simply upload a scholarly document and the metadata will be automatically extracted using Wiiliams et al. [36] to be used in the underlying algorithm. In the backend this API runs a disambiguation service with a faster query latency and at the frontend is a simple interface to provide easy access for the users.

1.3 Scope of Thesis

The success of any method that aims to provide an optimal solution to a problem is measured in terms of the accuracy and latency to provide the results. In this thesis, we focus on the finding a high quality adaptive querying approach for disambiguated author names in academic digital libraries. More specifically, we use the querying approach described in Treeratpituk and Giles [33], to compare with a more hierarchical approach which offers a faster latency with accuracy comparable to the former approach. This thesis has the following main contributions:

1. We propose to substitute the use of the random forest classifier with a new distance measure.

2. We explore various statistics to replace the random forest classifier and finally settle on calculating the median of the feature vector as the new distance measure. We evaluate and compare my model with other statistic such as mean on the Pubmed digital library. For Pubmed, we construct a new author disambiguated testbed which is used for experimenting and comparison purposes. Our experiments show that the proposed hierarchical model achieves significantly better pairwise disambiguation accuracy over the random forest approach.

3. Through variable analysis and feature selection, we identify a small set of features that can achieve high performance with a faster latency. At the second level, to maintain the accuracy of the previous approach, only for the mis-classified clusters, we use random forest classifier, of the previous method, thus ensuring the maintenance of accuracy. Our experiments show that the hierarchical model achieves significantly accurate and faster results.

The rest of this thesis is organized as follows. Chapter 2 describes related works in author name disambiguation, gives a brief description of the Pubmed digital libraries that we used in our evaluation. It also highlights the importance of variables and decribes the algorithm for querying disambugated databases using Random Forests. Chapter 3 presents our proposed approach and the algorithm for uerying disambugated databases using Median as a measure. This algorithm forms the backend of our API. Chapter 4 presents and discusses our evaluation results. Chapter 5

provides a summary of this thesis.

Chapter 2
Background

2.1 Related Work

2.1.1 Supervised and Unsupervised Methods

Past work on name disambiguation can for the most part be arranged as either supervised or those based on heuristics, unsupervised methodologies. Bekkerman et al. in [2] give an example of ambiguity in a query subject. This ambiguity consits of the same name shared by multiple people, an example highlighted in the previous chapter. The authors address this problem by presenting two statistical frameworks, one based on the link structure and the other on likeliness between topic-distribution which constitue as unsupervised methods.

In comparison, supervised methods learn the similarity from labeled examples [16, 28, 32, 33]. Methods that use both supervised and unsupervised techniques are called Hybrid methods. Torvik et al. [32] and Ferreira et al. [14] utilize heuristics to naturally create reference sets and utilize them, rather than named illustrations, as training information for the supervised techniques. In Santana et al. [27] disambiguation is led utilizing heuristics, with supervision being connected to improve the heuristics parameters. Thus combining both the supervised and unsupervised techniques.

Individual name disambiguation strategies can be classified in view of the sorts of data they utilize. In the digital libraries, the frequently utilized data is found in citations, for example, titles, coauthors, and venues [16, 28]. In addition to this frequently used data, Han et al. [16, 33] use affiliations and Song et al. [28] use abstracts. Moreover, these affiliations and abstracts can be extracted from the papers themselves. On the other hand Bekkerman et al. [2] use data such as citation graphs and co-authorship graphs. Pereira et al. [24] use external information sources for extracting data, such as search engines and author homepages for disambiguation.

In comparision to supervised methods, there is not much light shed on unsupervised methods that use clustering techniques especially for author name disambiguation. To bridge this gap, Ester et al. introduced an incremental version of the DBSCAN algorithm [11]. On the other hand Cao et al. introduce “DenStream”, a new approach for discovering clusters in an evolving data stream [7]. Wagstaff and Cardie propose two types of instance-level clustering constraints,

must-link and cannot-link [35]. Carlos Ruiz et al. introduce C-DBSCAN to accommodate for constraints when clustering data [26]. However, their work uses instance level constraints only such as must-link and cannot-link.

The objective of entity resolution is to merge different records that point to the same element together so as to reduce data duplication. In the case of author name disambiguation, these records are the metadata of a scholarly document and the entity is an author. Entity resolution is well studied in many different research communities under multiple names. It is referred to as record linkage [13], database hardening [10], and duplicate detection [37] in the database community. The Natural Language Processing community refers to it as coreference resolution [29] while other names include string matching [5].

2.1.2 Linkage Functions and Blocking Mechanisms

Much research in the area of author name disambiguation has focused on the accuracy of linkage function and a blocking mechanism [2, 5, 9, 19, 33]. The research of linkage functions ranges from using simple string editing distance to machine learning techniques such as decision trees. Tejada et al. used decision tree to develop an object identification system called Active Atlas, which compares the objects′ shared attributes in order to identify matching objects [31], while Han et al. proposed two classifiers, hybrid Naive Bayes and Support Vector Machine (SVM), for disambiguating authors in the computer science bibliography website, DBLP [11].

Blocking Mechanisms, also called as black-box linkage function focused on improving efficiency. The major advantage of using blocking functions is that the search space is reduced by a considerable amount. If all the data is organized into blocks, each record is only only compared with the records in the block and not all the data cumulatively. Thus the search space is reduced. Further this facilitates improvement in efficiency since now the pairwise comparisons can be reduced to O(nb) where n is the number of blocks and b is the size of a block as compared to $O(n^2)$, the complexity when no blocking mechanism is implemented. [3, 4, 17, 18, 20, 23]

2.1.3 PubMed

PubMed is a free search engine accessing primarily the MEDLINE database of references and abstracts on life sciences and biomedical topics. Medline is a de facto literature resource for the biomedical research. It contains journal citations and abstracts for biomedical literature from around the world. It has metadata of over 16 millions articles dating from 1965, including 4,500 scientific journals from over 80 countries. In addition to Medline, PubMed provides access to older references from back to 1951 and earlier references to some journals before they were indexed in Index Medicus and MEDLINE, for instance, Science, BMJ, and Annals of Surgery. It further provides access to very recent entries to records for an article before it is indexed with Medical Subject Headings (MeSH) and added to MEDLINE, a collection of books available full-text and other subsets of NLM records [12] and PMC citations.

As of 5 January 2017, PubMed has more than 26.8 million records going back to 1966, selectively to the year 1865, and very selectively to 1809; about 500,000 new records are added

(1)	Article Title	Selectivity: Who really should be offered ablations
	Affiliation	
	Authors	J Chen
	Journal Title	J Cardiovasc Electrophysiol
	Pub Date	2017 ay 9
	Mesh Heading	clinical: catheter ablation - atrial fibrillation
(2)	Article Title	Differential phenotypes of memory CD4 and CD8 T cells in the spleen and peripheral tissues following immunostimulatory therapy.
	Affiliation	Department of Dermatology, University of California, Davis School of Medicine
	Authors	Sckisel GD, Mirsoian A, Minnar CM, Crittenden M, Curti B, Chen JQ, Blazar BR
	Journal Title	Journal for immunotherapy of cancer
	Pub Date	2017 Apr 18
	Mesh Heading	Bystander Activation, CD8, Cancer, Immunotherapy
(3)	Article Title	Inadequate Systems to Support Breast and Cervical Cancer Screening in Primary Care Practice.
	Affiliation	Brigham and Women's Hospital
	Authors	Schapira MM, Sprague BL, Klabunde CN, Tosteson AN, Bitton A, Chen JS
	Journal Title	Journal of general internal medicine
	Pub Date	2016 Oct;31
	Mesh Heading	breast cancer screening, cervical cancer screening, patient-centered medical home.

Table 2.1. Examples of metadata in three Pubmed papers authored by a unique author.

each year. As of the same date, 13.1 million of PubMed's records are listed with their abstracts, and 14.2 million articles have links to full-text [1].

Table 2.1 shows examples of three articles in Pubmed written by author with the name "Chen J".

In Table 2.1, all the three articles contain the name "Chen, J". For articles (2) and (3), it is difficult to determine whether they are both written by the same "Chen, J.", (2)′s affiliation is Department of Dermatology, University of California, Davis School of Medicine and (3)′s affiliations is Brigham and Women's Hospital. There are clues, however, that suggest the possibility of (2) and (3) being a match. Both mesh headings are related to "cancer". For the articles (1) and (3), they are highly likely to be written by the same "Chen, J." Not only is the affiliation of (1) missing, but also both articles may not be about cancer. This unsureity and ambiguity is the major concern addressed by author name disambiguation techniques.

2.2 Disambuating Algorithm using Random Forests

In this section the algorithm for "Querying Disambugated Databases using Random Forests", inspired from Treeratpituk and Giles [33] which forms the basis for the algorithm developed by this thesis is decribed. First the random forest classifier is explained, then the definition of variable importance, and how variable importance can be inferred is explained. Finally the the feature set for disambiguating author names in Pubmed is defined. As mentioned earlier, since it assumed that the clusters are already disambiguated the algorithm section just describes

[1] `https://www.ncbi.nlm.nih.gov/pubmed/`

the querying process used to select the appropriate cluster for a given user query. The entire disabmiguation algorithm is stated and described in Appendix A.

2.2.1 Random Forests

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees developed by Breiman [6]. Each decision tree within the forests 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 as shown in Figure 2. Once the forest is built, the classification can be done by simply aggregating the votes of all trees.

A Decision Tree, while it has a low bias, it generally suffers from high variance leading to a high error rate. It is usually susceptible to noise in the data. A slight change in the training data often affects the structure of the tree dramatically. In comparison, Random forests gain their performance improvement over decision trees by achieving both low bias and low variance. It accomplishes this by aggregating a large number of low-correlated decision trees, each of which has low bias and high variance. The low bias of a forest is achieved by growing each tree without any pruning. The low variance is obtained from bagging (bootstrap aggregating) and random variable selection. Random forests have been reported to achieve performance that is even better than that of SVMs over a wide range of classification problems [6].

2.2.2 Variable Importance

A random forest offers a simple way to measure variable importance for each of its features, giving insights into the interaction between each feature and the prediction accuracy. This makes random forest attractive compared to other classifiers such as kernel-based SVM, which though often achieves good classification error rates, is hard to interpret.

The influence a parameter has on the prediction accuracy is measured by Variable importance. The method used for measuring variable importance in a random forest is: by Gini importance and by permutation importance. Gini importance is calculated based on Gini Index (or Gini Impurity), which is the measure of class distribution within a node. Gini index of a node i, IG(i), is defined as:

$$IG(i) = 1 - \sum_{j=1}^{K} pj^2 \tag{2.1}$$

where pj is the proportion of instances of class j in the node i, and K is the number of classes. IG(i) is minimum (= 0) when the node is pure. Gini index is used as the criteria for selecting the split at each node in the decision tree construction; the split that yeilds the biggest reduction in Gini index is selected. Therefore, in a decision tree, Gini impurity of the two descendent nodes is always less than that of the parent. Then, Gini importance of a variable can be computed by

averaging the Gini decreases for that variable over all trees in the forest. A variable with high Gini importance is the one that on average provides informative partitioning of data.

2.2.3 Feature Set

The similarity function used in this thesis is based of of, the features implemented by Treeratpituk et al. [33] and are as described below. However, I do not use all the features implemented by the former paper, rather decided by the variable importance I use a set of 15 features.

2.2.3.1 Input Vector

Given two papers, paperA and paperB, both containing an author with the name "lname, init" where lname refers to the last name and init, the first initial, the objective is to disambigute whether they refer to the same person. A naive blocking function that blocks author name based on the last name and the first initial is used. This is a reasonable assumption for a manually created database such as Pubmed, where the errors in the author names are low.
The following metadata of paperA and paperB are used to construct the input vector:

paperA = (lnameA,fnameA,midA,coauthA, affA, titleA, jourA, langA, yearA, meshA)
paperB = (lnameB,fnameB,initB,midB,sufB, coauthB, affB, titleB, jourB, langB, yearB, meshB)

where the terms are denoted as:

- lname(i) = the author′s last name in paper i
- fname(i)= the author′s first name in paper i
- mid(i) = the author′s middle name in paper i, if given
- coauth(i) = set of coauthors′ last name in paper i
- aff(i) = affiliation of the paper i′s 1st author
- title(i) = paper i′s title
- jour(i) = paper i′s journal name
- lang(i) = paper i′s journal language e.g. English, Chinese
- year(i) = paper i′s year of publication
- mesh(i) = set of mesh terms in the paper i

2.2.3.2 Pairwise Feature Vector

The feature set between author name in paperA and paperB consists of 15 features, which can be grouped into six categories based on the metadata, they are calculated from: author similarity, affiliation similarity, coauthors similarity, concept similarity, journal similarity, and title similarity. The 15 features are defined as follows:

Author Similarity

- auth_fst: the first name similarity.

$$auth\_fst = \begin{cases} 0 & \text{if } fname_A \neq fname_B, \text{both are given} \\ 1 & \text{if one of the first names is missing, and } init_A \neq init_B \\ 2 & \text{if one if the first names is missing, and } init_A{=}init_B \\ 3 & \text{if } fname_A{=}fanme_B, \text{both are given} \end{cases} \tag{2.2}$$

- auth_mid: similarity between midA and midB.

$$auth\_mid = \begin{cases} 0 & \text{if } mid_A, mid_B, \text{are given, } mid_A \neq mid_B \\ 1 & \text{if both } mid_A, mid_B \text{ are not given} \\ 2 & \text{if only one of } mid_A, mid_B \text{ are given} \\ 3 & \text{if } mid_A, mid_B \text{ are given, } mid_A{=}mid_B \end{cases} \tag{2.3}$$

- auth_lname_idf: IDF weight of the author last name. IDF is the inverse of the fraction of names in the corpus.

$$auth\_lname\_idf = log(IDF(lnameA)) \tag{2.4}$$

where the I D F is defined as:

$$IDF(lnameA) = IDF(lnameB) = \frac{L}{DF_{lname_A}} \tag{2.5}$$

and L is the total number of articles in Pubmed, $\mathrm{DF}_{\mathrm{lname_A}}$ is the total numbers of lnameA in Pubmed. High value of IDF(lname) means that lname is rare, thus is less likely to be ambiguous.

- auth_ord: similarity between the orders of author.

$$auth\_ord = \begin{cases} 2 & \text{if both authors are the 1st author} \\ 1 & \text{if both authors are the last author} \\ 0 & \text{otherwise} \end{cases} \tag{2.6}$$

Affiliation Similarity

- aff_jac: the jaccard similarity between aff_A and aff_B

$$aff\_jac = \frac{|aff_A \cap aff_B|}{|aff_A| + |aff_B|} \tag{2.7}$$

Coauthors Similarity

- coauth_lname_shared: the number of shared coauthor last names between the two papers.

$$coauth\_lname\_shared = |coauth_A \cap coauth_B| \tag{2.8}$$

- coauth_lname_idf: the sum of IDF values of all shared coauthor last names.

$$coauth\_lname\_idf = \sum_{l_n \in coauth_A \cap coauth_B} log(IDF(ln)) \tag{2.9}$$

- coauth_lname_jac: the jaccard similarity between coauth_A and coauth_B

$$coauth\_lname\_jac = \frac{|coauth_A \cap coauth_B|}{|coauth_A| + |coauth_B|} \tag{2.10}$$

Concept Similarity

- mesh_shared: the number of shared mesh terms between the two papers.

$$mesh\_shared = |mesh_A \cap mesh_B| \tag{2.11}$$

- mesh_shared_idf: the sum of IDF values of all shared mesh terms.

$$mesh\_shared\_idf = \sum_{t \in mesh_A \cap mesh_B} log(IDF(t)) \tag{2.12}$$

Journal Similarity

- jour_shared_idf: IDF value of the shared journal, if both are published in the same journal. The less common the journal is, the more informative this feature should be.

$$jourjour\_shared\_idf = \begin{cases} log(IDF(jourA)) & \text{if } jour_A = jour_B \\ 0 & \text{otherwise} \end{cases} \tag{2.13}$$

- jour_year: categorical variable reflecting change in PubmedâĂŹs policy.

$$jour\_year = \begin{cases} 0 & \text{if both are before 1988} \\ 1 & \text{if one is before 1988, one is after 1988} \\ 2 & \text{if both are between 1988 and 2002} \\ 3 & \text{if both are after 2002} \end{cases} \tag{2.14}$$

- jour_year_diff: difference in publication years.

$$jour\_year\_diff = |year_A - year_B| \tag{2.15}$$

Title Similarity

- title_shared: the jaccard similarity between title_A and title_B.

$$title\_shared = \frac{|title_A \cap title_B|}{|title_A| + |title_B|} \tag{2.16}$$

2.2.4 Algorithm

Algorithm 1 Querying Disambugated Databases using Random Forests

Require: The User′s query (i)
Ensure: $i \rightarrow nearest\_cluster$
  $AuthorBlock \leftarrow$ BlockingFunction(i)
  $count = int$[number of clusters in AuthorBlock]
  for all author mentions (j) in AuthorBlock **do**
    similarity $\leftarrow$ Similarity Function(i,j)# The Similarity Function is computed using the RF classifier
    if $similarity \leq \epsilon$ **then**
      $k \leftarrow$ clusterid that belongs to j
      $count[k] \leftarrow count[k] + 1$
    end if
  end for
  find clusterid that has maximum count
  return cluster

The user′s query is the input to the algorithm, while the desired output is the cluster to which the query belongs to. Depending upon the user′s query, a Blocking mechanism determines which block is chosen according to the lastname and first initial of the user query. The similarity function as described above is then computed for the query and all the author points in the chosen block which is then fed into the random forest classifier. The random forest then determines the probability of similarity between the query and all the other author points within the block. A clustering approach is then applied by computing the ε neighborhood of the user query. The ε neighborhood of a record r is a set of records with distance from r less than ε value. Here the value of ε chosen is 0.1. Thus from this ε neighborhood, for the record that has the least distance,

the cluster to which this record belongs to is chosen as the representative cluster of the user′s query.

2.3 Previous Works related to APIs for Name Dismabiguation

As mentioned in Section 2.1, various authors have presented varied methods of solving the name disambiguation problem. However, these approaches lack an interface with with each users can access the desired results with the click od a button. This section is thus focused on describing methods that provide an end-to-end service by providing an online service to facilitate user experience and satisfaction.

Yosef et al. [38] created an online tool for entity detection and disambiguation. This paper uses a linkage function by tranforming it into a graph problem where mentions from the input text and candidate entities define the node set, the weighted edges between mentions and entities capture context similarities, and weighted edges among entities, capture coherence. Usbeck et al. [34], uses string similarity measures, an expansion heuristic for labels to cope with co-referencing and the graph-based Hypertext-Induced Topic Search (HITS) algorithm for the disambiguation.

Ceccarelli et al. [8] is an open-source implementation of an entity disambiguation framework. The system was implemented in order to simplify the implementation of an entity linking approach and allows to replace single parts of the process. The authors implemented several state-of-the-art disambiguation methods. The entity linking task in this paper has been performed in three steps: 1. Spotting: a user mention is detected in the input document, and for each mention a list of candidate entities is retrieved. 2. Disambiguation: for each spot associated with more than one candidate, a single entity is selected to be linked to the spot. 3. the entities thus detected are then ranked by some policy.

Many entity linking algorithms and entity disambiguation algorithms have been proposed, but unfortunately only a few authors have released some APIs which provides the motivation for the design, development and implementation of an API.

Chapter 3
Proposed Approach

The success of any method that aims to provide an optimal solution to a problem is measured in terms of the accuracy and latency to provide the results. The previous approach of author name disambiguation using random forest is computationally intensive with lower query latency. Query latency here refers to the time taken by the algorithm to disambiguate and display accurate results for the user's query.

Thus, the aim of this thesis is to remove the component of random forest and employ another measure to compute the distance, thereby improving the query latency and without affecting the accuracy of clustering acieved by using the random forest classifier. The similar approach of computing the author blocks by using a blocking mechanism of using the authors' last names and the first initial of the first nane, the variable importance and similarity function as shown in the previous section are adapted and explained below.

3.1 Querying disambiguated database using Median-based Method

In this section first the median distance function is explained, then the definition of variable importance, and how variable importance can be inferred is explained. Finally the the feature set for disambiguating author names in Medline is defined.

3.1.1 Using Median as the distance function

In the previous approach the random forest classifier is used to predict the distance between a pair of author records. This thesis has found that using Random Forest is found to be computationally intensive. It further claims that using the median, a measure of central tendency, as the distance between a pair of records reduces the amount of computation thereby with a faster query latency and with a similar accuracy to that of the random forest.

The overall complexity of random forests algorithm given in the previous Chapter 2 is:

$$complexity = O(T * M * (N) * log(N)) \tag{3.1}$$

where the terms are denoted as:

- T = number of trees
- M = the number of features to consider when splitting each node
- N = number of objects

For the following algorithm which shows only the computation of the Feature Set using Median:

Algorithm 2 Querying Disambugated Databases using Median

Require: The User′s query (i)
Ensure: $i \rightarrow nearest\_cluster$
  $AuthorBlock \leftarrow$ BlockingFunction(i)
  $count = int$[number of clusters in AuthorBlock]
  for all author mentions (j) in AuthorBlock **do**
    similarity ← Similarity Function(i,j)# The Similarity Function is computed using the Median
  end for

The complete algorithm is shown below in Section 3.1.6.

In comparison the overall complexity of median is:

$$O(nlog(n)) \tag{3.2}$$

where "n" is:

- n = number of objects

The difference between the two complexities lies in the number of trees parameter. The computation of median does not require the construction of trees to the maximum size. Further, it is found that computing the median between a pair of records takes lesser time in comparison to the random forest prediction function which predicts the pairwise distance between the records.

Further in this thesis the coefficient of correlation between the random forest technique and median techinque is shown in Chapter 4 which shows the linear relationship between the two methods.

3.1.1.1 Median and its Advantage

The median is a measure of central tendency. The median on the other hand is the value separating the higher half of a data sample, a population, or a probability distribution, from the lower half. The median is not skewed so much by extremely large or small values. If the data is skewed median is a better choice of statistic. One of the author blocks is represented by a graph as shown above in Figure 3.1. One can see from the figure above that the graph the skewed towards the left which advocates the use of median as a statistic in comparison to the use of median.

Further, the coefficient of correlation between the mean and randon forest techniques and the median and random forest techniques is shown in Chapter 4 to show a comparison between the choice of median vs the choice of mean as the distance measure between a pair of author records.

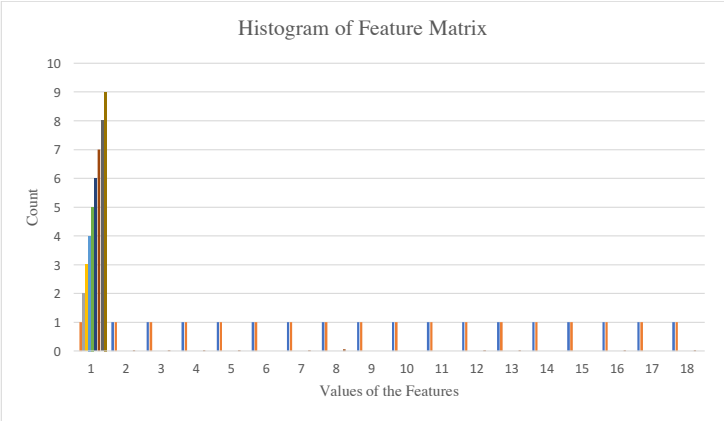

Figure 3.1. Histogram of Feature Matrix

auth_fst	auth_mid	auth_last_idf	auth_ord	year
0.37082862	0.24135436	0.03386861	0.00669803	0.01592611
year_diff	mesh_shared	mesh_shared_idf	title_shared	aff_jac
0.03153586	0.03867417	0.11925425	0.05141267	0.0045311
aff_jw	coauth_shared	coauth_idf	coauth_jac	jour_shared_idf
0.00338945	0.01135948	0.03661643	0.0220105	0.01254037

Table 3.1. Importance value of features

3.1.2 Variable Importance

In this section, the variable importance is calculated by the Gini Index offered by the random forest. It is represented as equation 2.1 in Chapter 2.

The higher the importance value the higher the variable affects the disambiguation performance and cannot be ignored in computing the input vector and the lower the importance is, the respective variable can be ignored without a considerable affect on the disambuguation process.

The gini importance is shown in Table 3.1.

As seen from Table 3.1, the values of variables auth_fst, auth_mid, auth_last_idf, year, year_diff, mesh_shared, mesh_shared_idf, title_shared, coauth_idf, coauth_jac are higher as compared to the remaining variables and thus the latter variables with lower values are not considered in the computation of the similarity function explained in the section below.

3.1.3 Feature Set

The Feature Set computed is based on the importance on the Variable importance as shown in Table 3.1. The feature set between author name in paperA and paperB consists of 10 features,

which can be grouped into six categories based on the metadata, they are calculated from: author similarity, affiliation similarity, coauthors similarity, concept similarity, journal similarity, and title similarity as described in detail in Chapter 2.

3.1.3.1 Input Vector

The following metadata of paperA and paperB are used to construct the input vector:

paperA = (auth_fstA, auth_midA, auth_last_idfA, yearA, year_diffA, mesh_sharedA, mesh_shared_idfA, title_sharedA, coauth_idfA, coauth_jacA)

paperB = (auth_fstB, auth_midB, auth_last_idfB, yearB, year_diffB, mesh_sharedB, mesh_shared_idfB, title_sharedB, coauth_idfB, coauth_jacB)

3.1.4 Improvement in Accuracy

In this section the approach using median as the distance is comapred with the approach of using random forest to predict the distance. This observation is shown in the Table 3 below. From Table 3 we can see that using the median to compute the pairwise distance between author records, the number of misclassified records increases and the conclusion reached is that just using the median to compute the pairwise distance between author records and form clusters belonging to a particular author is not sufficient. To overcome this and improve the classification and clustering of records so that the appropriate cluster can be displayed against the user's query we follow the following algorithm.

The median is first computed between the record belonging to the user's query and the remaining records within the particular author block. After computing the median, if the query is misclassified, the top k clusters to which the query could belong are obtained. The distance between the record belonging to the user's query and the records in these k clusters is then computed by the random forest. As shown in the previous section the accuracy of random forest as compared with using median is better. Thus here, the random forest is used to compute the distance. The results of this algorithm are shown in Chapter 4.

3.1.5 Algorithm

The user′s query is the input to the algorithm, while the desired output is the cluster to which the query belongs to. Depending upon the user′s query, a Blocking mechanism determines which block is chosen according to the lastname and first initial of the user query. The similarity function as described above is then computed for the query and all the author points in the chosen block which is then fed into the function that computes the Median. This function then computes the median which determines the probability of similarity between the query and all the other author points within the block. A clustering approach is then applied by computing the ε neighborhood of the user query. The ε neighborhood of a record r is a set of records with distance from r less than ε value. Here the value of ε chosen is 0.1 as in Khabsa et al. [19]. Thus

Algorithm 3 Querying Disambugated Databases using Hierarchical Approach

Require: The User′s query (i)
Ensure: $i \rightarrow nearest\_cluster$
  $AuthorBlock \leftarrow$ BlockingFunction(i)
  $count = int$[number of clusters in AuthorBlock]
  for all author mentions (j) in block **do**
    similarity ← Similarity Function(i,j) # The Similarity Function is computed using the Median
    if $similarity \leq \epsilon$ **then**
      $k \leftarrow$ clusterid that belongs to j
      $count[k] \leftarrow count[k] + 1$
    end if
  end for
  find clusterid that has maximum count
  if $i \neq k$ **then**
    $k \leftarrow$ top k clusterids that belongs to j
    $count[k] \leftarrow count[k]$+1# for all top k clusters
  end if
  find clusterid that has maximum count
  return cluster

from this ε neighborhood, for the record that has the least distance, the cluster to which this record belongs to is chosen as the representative cluster of the user′s query. However, if the query is misclassified, the top k clusters are then computed for this query. The top six clusters are computed using the above algorithm since it is found that the appropriate cluster to which the query actually belongs to appears within the top six clusters. For all the records in these six clusters, the feature set and input vector are then computed between the user query and these records. This output is then fed into the Random Forest classifier which predicts the probability of similarity between the query and all the other records in these top six clusters. The grouping approach is then repeated to find the appropriate cluster to which the user query belongs to.

3.2 Development of API

In order to provide an end-to-end service, a Web-accessible API can provide a single point of operation for disambiguation that can be integrated into any work that requires author name disambiguation without the need to implement and support its own disambiguation functionality.

3.2.1 Architecture

This API is a RESTful Web service based on the Resource Oriented Architecture (ROA) [25]. The advantages of RESTful Web services are plentiful, such as being lightweight, scalable and easily accessible [20]. ROA can further be defined by four main concepts: resources, identifiers, representations of resources, and the links between resources. The four main properties of ROA are: addressability, statelessness, connectedness, and a uniform interface [17].

A. The 4 moving parts of the Resource Oriented Architecture are leveraged in this API development in the following manner:

< ?xml version="1.0" encoding="UTF-8"?>
< CSXAPIMetadata> (**the unique identifier**)
< file> base url/extractor/token/file< /file>
< header> base url/extractor/token/header< /header>
< citations> base url/extractor/token/citations< /citations>
< body> base url/extractor/token/body< /body>
< text> base url/extractor/token/text< /text>
< /CSXAPIMetadata>

Table 3.2. "CSXAPIMetadata" is the Identifier

1) Resources: Resources are things that can be referenced themselves. Scholarly documents (PDFs), user input, are resources in this API since they are being directly referenced in order to extract information from them. The automatic extraction of metadata from the document resources is done by levaraging the CiteSeerExtractor [36]. The CiteSeerExtractor has a URL by which metadata information extracted from the resources can be accessed. Resources are created by submitting a POST request to this extractor URL. This has the effect of creating a new document resource in CiteSeerExtractor. The successful creation of a new resource through the submission of a PDF and the successful extraction of the text is identified by a HTTP 201 CREATED status code, whereas if an error occurs a HTTP 503 INTERNAL ERROR status code is returned.

2) Identifiers: Once a resource has successfully been created and the 201 code is returned, it is assigned a unique and random identifier. Figure 1 shows an example, taken from the identifier assigned by the CiteSeerExtractor, of the identifier for a new resource (the string of characters trailing extractor/ in the URL). This identifier uniquely identifies the new resource for as long as it exists and allows for representations to be extracted.

3) Representations: Representations of a resource are different views of a resource and in this API represents the particular cluster to which the resource could belong to.

4) Addressability, Statelessness, Connectedness, and Uniaformity: Representations of resources in this API are addressable through XMl or JSON output. Resources remain addressable for as long as the system retains the stored document. This is stateless as each HTTP request happens independently and is not dependent on any preceding requests. Connectedness is provided through links to representations of resources when a new resource is created through the CiteSeerExtractor.

B. HTTP Methods

Table 4 summarizes the HTTP methods supported by the API. As can be seen from the table, the first method involves using a HTTP POST to create a resource. Different representations of a resource can then be retrieved with a HTTP GET. Lastly, resources can be deleted with a HTTP DELETE on the representation URI. These methods capture most of the functionality that one would expect when disambiguating with information from scholarly documents. Furthermore, the API also supports different output formats, i.e., XML and JSON, for the returned data which can be easily used by users.

Method	URL	Description	Returns	Options
POST	/	Uploads a new PDF document via a form(if using CiteSeerExtractor) POSTS user data or BibTex via a form	XML document with URIs to resource, resource id	Required upload of a PDF file only or manual entry of user input
GET	/get_info	Used to display the cluster that contains documents belonging to input author	Representation of cluster information	Output=xml (default) | json
DELETE	/cluster	Deletes the resource	Confirmation	N/A

Table 3.3. HTTP methods supported by the API

3.2.2 API Design

The overall design of the API is designed to be stand-alone, able to run in isolation, and able to be integrated with a number of services. Figure 3.2 shows the overarching architecture of the API. As can be seen from the figure, the RESTful API is the entry point and communicates directly with the Python Flask Web Server, which is responsible for displaying the results. Security and permissions can also be controlled and implemented at the Python Flask Web Server level. In the rest of this section, the technology and implementation details for each level of the design are described.

A. RESTful API:

The RESTful API provides the functionality as described in the previous subsection.

B. Python Flask Web Server:

This API is run as a stand-alone Web server and is implemented using the Flask framework. The Flask server is responsible for the creation and removal of resources and handling all HTTP requests. The actual disambiguation functionality is provided by the above algorithm that is executed by the Flask server.

C. Header Extractor:

A support vector machine aids in header extraction of the document [15]. As with citation extraction, the section of text containing the header is identified using a regex and then each line of the header is classified and various aspects of the header, such as authors, title, etc., are classified and returned.

D. Hierarchical Algorithm:

This forms the backend of the API and is the Hierarchical algorithm explained in detail in Section 3.1.5.

E. Cluster:

The output of this API is a cluster of documents and articles belonging to the author name in the user query. The cluster consits of exactly those documents that match and point to the particular user query.

3.2.3 Backend

At the backend, this API used the above hierarchical algorithm. The input for this API, used by the algorithm at the backend can be of the following 3 forms: 1. Automatic Extraction done by the CiteSeerExtractor.
2. Manually entered metadata.
3. Manually entered data in the BibTex format.

This data is then used by the hierarcical algorithm, which is then processed and the appropriate cluster are then displayed in either XML or JSON format. This constitues the output of the API. The entire design of the API is as represted in Figure 3.2

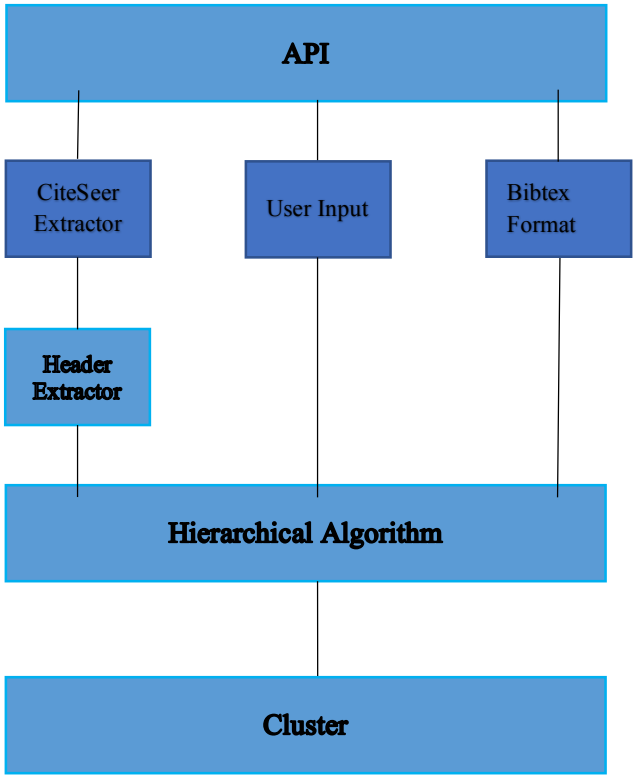

Figure 3.2. Design of the API

Chapter 4
Results and Experiments

4.1 Experiments

To evaluate the performance of the proposed approach that employs median in comparison to approach that uses the random forest classifier, the dataset used is the "NIH PI dataset" that has list of sampled PIs and their Pubmed publications as shown in Table 4.1. For this dataset, we sampled some blocks based on "Last name + First name initial", which is the blocking mechanism applied here. For each author record in this block, the articles are manually clustered. Each resulting cluster corresponds to the set of articles written by one unique author. These are used as the standard in the training and the evaluation. For the block, 100 author records are then selected at random as user queries. These are queries for which the appropriate and accurate corresponding cluster should be displayed.

In the method that uses random forest, after computing the similarity function between each user query and the rest of the records in the block, a feature vector is obtained. The feature vector

pid	aid	first name	last name	title
15495495	46123869	Marion	Johnson	Content validity and nursing sensitivity of community-level outcomes from the Nursing Outcomes Classification (NOC)
12930463	43110779	Marion	Johnson	Validity and community-health-nursing sensitivity of six outcomes for community health nursing with older clients.
11775308	39371209	M	Johnson	Genetic counseling outcomes validation by genetics nurses in the UK and US.
6307410	13578134	M E	Johnson	Saturation transfer electron paramagnetic resonance detection of sickle hemoglobin aggregation during deoxygenation.
6838898	13405032	M E	Johnson	Probable binding region of small hydrophobic molecules on hemoglobin. Spin label-induced nuclear magnetic relaxation.
8441811	24633020	M E	Johnson	Derivation of locally accurate spatial protein structure from NMR data.
1445211	24006291	M E	Johnson	Bivalent-metal binding to CheY protein. Effect on protein conformation.
1542119	23135205	M E	Johnson	Location of potential binding sites on deoxy hemoglobin for the design of antigelling agents.

Table 4.1. Some rows of Author Block Data

Random Forest Approach	Median Approach
1.14	1.4

Table 4.2. Difference in mean rank

is then used by the random forest classifier to predict the probability of which cluster the query could belong to. In comparison in the proposed approach, after computing the similarity function between each user query and the rest of the records in the block, a feature vector is obtained. The median of each vector is then computed which is then used to display the appropriate cluster for each query.

4.1.1 Coefficient of Correlation

Correlation presents a mutual relationship between two parameters and coefficient of correlation is a quantity measuring the extent of interdependence of variable quantities. Computing the coefficient of correlation is thus very vital to support the decision of using the median in comparison to using the random forest classifier.

The coefficient of correlation between the two methods obtained is 0.998. This suggests a very strong co-relation, which shows a very positive linear relation between the two methods and advocates the use of median as a statistic for distance computation.

As mentioned in Chapter 3, the hierarchical approach applies the RF classifier method only for queries which are mis-classified at the first level. This RF classifier method performs pairwise comparisons between the mis-classified query record and the records in the top k clusters to which the query could have belonged to. In our algorithm we consider the top 6 clusters as mentioned in Chapter 3.

Thus, at the first level, for the mis-classified queries, it is important to compare the presence of the appropriate cluster in the top 6 clusters, whether it is present or not. The Table 4.1 shows this comparison between the two methods.

From Table 4.2 we can see that in the Random Forest approach for the 100 queries we have a mean rank of 1.14. Similarly, for the median method, we have a mean rank of 1.4. Here we can see that the mean rank is better for the Random forest approach as compared to the median approach. Thus, through the hierarchical method we apply the Random Forest approach with an aim to better this mean rank. This result is documented in Table 4.7

Applying only the first level of the hierarchical approach, the median method, we thus provide a filtering approach which filter out the queries whose top 6 clusters have the approriate cluster ids and run the parwise comparisons using the random forest method on only these query records present in the 6 clusters.

4.1.2 Comparison of Query Latency

The previous chapter spoke about why median is used as the statistic and since the propsed approach is used to improve the query latency, the difference of time for distance computation

Random Forest Approach	Median Approach
15 seconds	0.9 seconds

Table 4.3. Time difference in computing the distance matrix

Random Forest Approach	Median Approach
76 percent (24 misclassified)	75 percent (25 misclassified)

Table 4.4. Difference in Accuracy for the classification of 100 random queries

between the 2 methods for the block data shown in Table 4.1 is shown in the Table 4.3:

As can be seen from the table above the difference in time between the two methods is 14 seconds which is a considerable amount.

4.1.3 Comparison of Accuracy for the Median method

The comparison of accuracy between the two methods is extremely vital since accuracy here refers to the query being properly classified so that the appropriate corresponding cluster is displayed to which the query actually belongs to. For the block in Table 4.1, the accuracy between both the methods is documented in the Table 4.4.

From Table 4.4, it can be seen that the accuracy of the random forest method is better as compared to the median approach. However, to improve this accuracy of the median approach we apply the second level of the hierarchical approach explained in the above algorithm. For the 25 mis-classified queries now the random forest classifier is applied to classify these queries to belong to their appropriate clusters. This certainly adds an overhead in terms of the query latency as shown in Table 4.5 , however the accuracy improves above that of the random forest method as shown in Table 4.6.

Further, from Table 4.4 we can see an increase in time taken, however including this increase, the hierarchical approach is still a faster method that takes 5 seconds to return the appropriate cluster ids for 100 queries as compared to the 15 seconds taken by the random forest approach.

4.1.4 Comparison of Mean Rank

As desribed above the mean rank calculates the appearence of the appropriate cluster ids, to which the 100 queries actually belong to, in the top 6 clusters. A mean rank of 1 indicates that the appropriate cluster id is the top hit for the particular query. Thus, higher mean rank indicates a greater mis-classification of queries and a lower mean rank indicates better classification.

Random Forest Approach	Hierarchical Approach
15 seconds	5 seconds

Table 4.5. Time difference in computing the distance matrix

Random Forest Approach	Hierarchical Approach
76 percent (24 misclassified)	80 percent (20 misclassified queries)

Table 4.6. Difference in Accuracy for the classification of 100 random queries

Random Forest Approach	Hierarchical Approach
1.14	1.10

Table 4.7. Difference in mean rank

As seen from table 4.6, the mean rank of the combined hierarchical approach is much lower than that of the Random forest approach.

The reason for improvement in query latency, accuracy and mean rank is mainly due to the hierarchical approach. The random forest method is known to have the best accuracy amongs other machine learning approaches as can be seen in [33]. However the computation and generation of results is much slower as compared to the hierarchical approach.

Thus the hierarchical approach is much faster and maintains the accuracy of the random forest approach.

Chapter 5
Conclusion

5.1 Summary

This thesis demonstrate how a hierarchical approach can be adapted to the problem of author name disambiguation in scientific databases. The hierarchical approach, at the first level, consists of substituing the use of random forests with median as a statistic to disambiguate and classify a user query. the second level takes place only if the query is mis-classified. Then at this second level, the disambiguation takes place using random forest. This hierarchical approach thus ensures a faster query latency while maintaining the accuaracy, if not improving it. The end result of this thesis is an end-to-end service with a RESTful API which uses the hierarchical algorithm at its backend.

5.2 Future Research

For future work, there are multiple things that we would like to do. First, we plan to perform an even faster way of querying using median as the distance measure by reducing the number of pairwise comparisons to just one. Second we would like to integrate this hierarchical approach with other clustering approaches as a means of comparison and third we woud like to extend the experimenting to other databases.

Appendix | Disambiguation Algorithm using Random Forest

Algorithm 4 Disambiguation algorithm using Random Forest

Require: author records in database D
Ensure: $clusters \leftarrow D$
  for all records within the database **do**
    Preprocessing
    BlockingFunction
    $BlockingFunction \rightarrow authorBlock$
    for all block in authorBlocks **do**
      $PairwiseClassification(paper_{query}, paper_{block})$
      $clusters \leftarrow DBSCANclustering(block)$
    end for
  end for
  return *clusters*

The above disambiguation algorithm has been adapted from [21, 33].

Algorithm 5 DBSCAN

Require: Static collection of records to be disambiguated
  mark all records in D as UNVISITED
  for all records p in D **do**
    if p is UNVISITED **then**
      mark p as VISITED
      $N \leftarrow query(D, p, \epsilon)$# the Random Forest based pairwse classifier is used to get the neighbors
      if $|N| \leq minPts$ **then**
        assign p $\rightarrow NOISE$
      else
        expandCluster(p,N)
      end if
    end if
  end for

The above DBSCAN algorithm has been adapted from [19]

Bibliography

[1] Javier Artiles, Julio Gonzalo, and Felisa 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*, pages 569–570. ACM, 2005.

[2] Ron Bekkerman and Andrew McCallum. Disambiguating web appearances of people in a social network. In *Proceedings of the 14th international conference on World Wide Web*, pages 463–470. ACM, 2005.

[3] Omar Benjelloun, Hector Garcia-Molina, Heng Gong, Hideki Kawai, Tait E Larson, David Menestrina, and Sutthipong Thavisomboon. D-swoosh: A family of algorithms for generic, distributed entity resolution. In *Distributed Computing Systems, 2007. ICDCS'07. 27th International Conference on*, pages 37–37. IEEE, 2007.

[4] Mikhail Bilenko, Beena Kamath, and Raymond J Mooney. Adaptive blocking: Learning to scale up record linkage. In *Data Mining, 2006. ICDM'06. Sixth International Conference on*, pages 87–96. IEEE, 2006.

[5] Mikhail Bilenko, Raymond Mooney, William Cohen, Pradeep Ravikumar, and Stephen Fienberg. Adaptive name matching in information integration. *IEEE Intelligent Systems*, 18(5):16–23, 2003.

[6] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[7] Feng Cao, Martin Estert, Weining Qian, and Aoying Zhou. Density-based clustering over an evolving data stream with noise. In *Proceedings of the 2006 SIAM international conference on data mining*, pages 328–339. SIAM, 2006.

[8] Diego Ceccarelli, Claudio Lucchese, Salvatore Orlando, Raffaele Perego, and Salvatore Trani. Dexter: an open source framework for entity linking. In *Proceedings of the sixth international workshop on Exploiting semantic annotations in information retrieval*, pages 17–20. ACM, 2013.

[9] Peter Christen. A comparison of personal name matching: Techniques and practical issues. In *Data Mining Workshops, 2006. ICDM Workshops 2006. Sixth IEEE International Conference on*, pages 290–294. IEEE, 2006.

[10] William W Cohen, Henry Kautz, and David McAllester. Hardening soft information sources. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 255–259. ACM, 2000.

[11] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Michael Wimmer, and Xiaowei Xu. Incremental clustering for mining in a data warehousing environment. In *VLDB*, volume 98, pages 323–333, 1998.

[12] Matthew E Falagas, Eleni I Pitsouni, George A Malietzis, and Georgios Pappas. Comparison of pubmed, scopus, web of science, and google scholar: strengths and weaknesses. *The FASEB journal*, 22(2):338–342, 2008.

[13] Ivan P Fellegi and Alan B Sunter. A theory for record linkage. *Journal of the American Statistical Association*, 64(328):1183–1210, 1969.

[14] Anderson A Ferreira, Adriano Veloso, Marcos André Gonçalves, and Alberto HF Laender. Effective self-training author name disambiguation in scholarly digital libraries. In *Proceedings of the 10th annual joint conference on Digital libraries*, pages 39–48. ACM, 2010.

[15] Hui Han, C Lee Giles, Eren Manavoglu, Hongyuan Zha, Zhenyue Zhang, and Edward A Fox. Automatic document metadata extraction using support vector machines. In *Digital Libraries, 2003. Proceedings. 2003 Joint Conference on*, pages 37–48. IEEE, 2003.

[16] Hui Han, Lee Giles, Hongyuan Zha, Cheng Li, and Kostas Tsioutsiouliklis. Two supervised learning approaches for name disambiguation in author citations. In *Digital Libraries, 2004. Proceedings of the 2004 joint ACM/IEEE conference on*, pages 296–305. IEEE, 2004.

[17] Mauricio A Hernández and Salvatore J Stolfo. The merge/purge problem for large databases. In *ACM Sigmod Record*, volume 24, pages 127–138. ACM, 1995.

[18] Liang Jin, Chen Li, and Sharad Mehrotra. Efficient record linkage in large data sets. In *Database Systems for Advanced Applications, 2003.(DASFAA 2003). Proceedings. Eighth International Conference on*, pages 137–146. IEEE, 2003.

[19] Madian Khabsa, Pucktada Treeratpituk, and C Lee Giles. Online person name disambiguation with constraints. In *Proceedings of the 15th ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 37–46. ACM, 2015.

[20] Hung-sik Kim and Dongwon Lee. Parallel linkage. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 283–292. ACM, 2007.

[21] Kunho Kim, Madian Khabsa, and C Lee Giles. Random forest dbscan clustering for uspto inventor name disambiguation and conflation. 2016.

[22] Shiyong Lu and Jia Zhang. Collaborative scientific workflows. In *Web Services, 2009. ICWS 2009. IEEE International Conference on*, pages 527–534. IEEE, 2009.

[23] Alvaro Monge and Charles Elkan. An efficient domain-independent algorithm for detecting approximately duplicate database records. 1997.

[24] Denilson Alves Pereira, Berthier Ribeiro-Neto, Nivio Ziviani, Alberto HF Laender, Marcos André Gonçalves, and Anderson A Ferreira. Using web information for author name disambiguation. In *Proceedings of the 9th ACM/IEEE-CS joint conference on Digital libraries*, pages 49–58. ACM, 2009.

[25] Leonard Richardson and Sam Ruby. *RESTful web services*. " O'Reilly Media, Inc.", 2008.

[26] Carlos Ruiz, Myra Spiliopoulou, and Ernestina Menasalvas. C-dbscan: Density-based clustering with constraints. In *International Workshop on Rough Sets, Fuzzy Sets, Data Mining, and Granular-Soft Computing*, pages 216–223. Springer, 2007.

[27] Alan Filipe Santana, Marcos André Gonçalves, Alberto HF Laender, and Anderson Ferreira. Combining domain-specific heuristics for author name disambiguation. In *Proceedings of the 14th ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 173–182. IEEE Press, 2014.

[28] Yang Song, Jian Huang, Isaac G Councill, Jia Li, and C Lee Giles. Efficient topic-based unsupervised name disambiguation. In *Proceedings of the 7th ACM/IEEE-CS joint conference on Digital libraries*, pages 342–351. ACM, 2007.

[29] Wee Meng Soon, Hwee Tou Ng, and Daniel Chung Yong Lim. A machine learning approach to coreference resolution of noun phrases. *Computational linguistics*, 27(4):521–544, 2001.

[30] Amanda Spink, Bernard J Jansen, and Jan Pedersen. Searching for people on web search engines. *Journal of Documentation*, 60(3):266–278, 2004.

[31] Sheila Tejada, Craig A Knoblock, and Steven Minton. Learning object identification rules for information integration. *Information Systems*, 26(8):607–633, 2001.

[32] Vetle I Torvik, Marc Weeber, Don R Swanson, and Neil R Smalheiser. A probabilistic similarity metric for medline records: A model for author name disambiguation. *Journal of the American Society for information science and technology*, 56(2):140–158, 2005.

[33] Pucktada Treeratpituk and C Lee Giles. Disambiguating authors in academic publications using random forests. In *Proceedings of the 9th ACM/IEEE-CS joint conference on Digital libraries*, pages 39–48. ACM, 2009.

[34] Ricardo Usbeck, Axel-Cyrille Ngonga Ngomo, Michael Röder, Daniel Gerber, Sandro Athaide Coelho, Sören Auer, and Andreas Both. Agdistis-graph-based disambiguation of named entities using linked data. In *International Semantic Web Conference*, pages 457–471. Springer, 2014.

[35] Kiri Wagstaff and Claire Cardie. Clustering with instance-level constraints. *AAAI/IAAI*, 1097, 2000.

[36] Kyle Williams, Lichi Li, Madian Khabsa, Jian Wu, Patrick C Shih, and C Lee Giles. A web service for scholarly big data information extraction. In *Web Services (ICWS), 2014 IEEE International Conference on*, pages 105–112. IEEE, 2014.

[37] Hui Yang and Jamie Callan. Near-duplicate detection for erulemaking. In *Proceedings of the 2005 national conference on Digital government research*, pages 78–86. Digital Government Society of North America, 2005.

[38] Mohamed Amir Yosef, Johannes Hoffart, Ilaria Bordino, Marc Spaniol, and Gerhard Weikum. Aida: An online tool for accurate disambiguation of named entities in text and tables. *Proceedings of the VLDB Endowment*, 4(12):1450–1453, 2011.