Palestine Polytechnic University
Deanship of Graduate Studies and Scientific Research
Master of informatics

A Heuristic-based Approach for Author Name Disambiguation

Submitted by:

Ibrahim M. Qdemat

Thesis submitted in partial fulfillment of requirements of the degree Master of Science in Informatics
July, 2017

The undersigned hereby certify that they have read, examined and recommended to the Deanship of Graduate Studies and Scientific Research at Palestine Polytechnic University the approval of the thesis entitled: **A Heuristic-based Approach for Author Name Disambiguation**, submitted by **Ibrahim M. Qdemat** in partial fulfillment of the requirements for the degree of Master in Informatics.

Graduate Advisory Committee:

Dr. Sami Taha Abu-Snaineh (Supervisor), Palestine Polytechnic University.

Signature:_____ Date:_____

Dr. Hashem Tamimi (Internal committee member), Palestine Polytechnic University.
Signature:_____ Date:_____

Dr. Beverly Jamison (External committee member), Practical Semantics.

Signature:_____ Date:_____

Thesis Approved

Dr. Murad Abu Sbeih Dean of Graduate Studies and Scientific Research Palestine Polytechnic University

Signature:_____ Date:_____

Abstract

Author name disambiguation is well known problem in digital libraries that aims at identifying the real owner of a scientific contribution. Author name disambiguation is a challenging problem because of different name strings ambiguities. For example, the same name might be written in many different ways. On the other hand, the same name string might be shared between different individuals. In this thesis, we propose a new approach to solve author name ambiguities. Our approach depends on a heuristic-based scoring method that utilizes different stages in an effort to take the disambiguation decision as early as possible. In addition, the algorithm is designed to be both scalable for large databases and adaptive based on the case in hand. The algorithm is validated against a wide variety of manually labeled datasets. Our results showed that about 91.03% of the generated profiles exactly matched the profiles in the reference datasets and about 8.18% partially matches the reference profiles and only less than 0.8% of error profiles. Moreover, we ran the algorithm against more than 10 million name string instances in real database within a relatively short time.

الملخص

تحتوي المكتبات الالكترونية هذه الأيام على كميات هائلة من الأوراق العلمية و تقدم خدمات كثيرة للباحثين من اهمها ما يتعلق باسماء المؤلفين، فعلى سبيل المثال يمكن البحث باستخدام اسم المؤلف و تقييم المؤلف باستخدام مقاييس مختلفة بناء على منشوراته العلمية وغيرها من الخدمات. من اهم المشكلات امام تحقيق هذه الخدمات هي مشكلة غموض الاسماء. فمن جهة، فانه من المحتمل أن ينشر المؤلف أوراقا علمية مختلفة مستخدما اشكالا مختلفة من اسمه كأن يبقي أو يحذف بعض الاجزاء أو يستخدم بعض الاختصارات. ومن جهة أخرى، فهناك عدد كبير من المؤلفين الذين يحملون نفس الاسم بشكل متطابق خصوصا أصحاب الاسماء المشهورة.

تهدف عملية ازالة الغموض عن الاسماء إلى تجميع المؤلفات التابعة لنفس الشخص حتى لو استخدم أشكالا مختلفة عند كتابة اسمه، وفي نفس الوقت تمييز المؤلفات العلمية للأشخاص الذين يحملون نفس الاسم عن بعضها البعض. في هذه الرسالة نقترح خوارزمية جديدة لإزالة الغموض عن الاسماء باستخدام الطرق الاستدلالية. لقد تم تصميم هذه الخوارزمية لتعمل بكفاءة و سرعة عالية من خلال تقسيم العمل الى مجموعة من المهام المستقلة وتشغيلها بشكل متوازي، بالاضافة الى قدرتها على العمل في مراحل بهدف اتخاذ القرار بأقل عدد ممكن من المقارانات وبأسرع وقت ممكن دون الحاجة لإجراء جميع الحسابات إن أمكن ذلك. كما أن الخوارزمية المقترحة تتمع بالقدرة على التكيف بناء على الحالة قيد المعالجة، فمثلا يتم التعامل مع الاسماء القصيرة والاسماء الصينية بطريقة مختلفة لانها اكثر تحديا وغموضا نتيجة التشابه الكبير بين الاسماء في هذه الحالات.

من أجل تقييم الخوارزمية المقترحة قمنا بتصنيف مجموعات مختلفة ومتنوعة من الأوراق العلمية بطريقة يدوية بناءً على المؤلف الحقيقي. أظهرت النتائج أن الخوارزمية المقترحة حققت دقة بنسبة تزيد بقليل عن 91% متطابقة تماما مع التصنيف اليدوي، ونسبة 8.18% متطابقة جزئيا مع التصنيف اليدوي، ونسبة خطأ تقل عن 91% كما أن الخوارزمية المقترحة يمكنها التعامل مع عدد هائل من الاسماء حيث تم تطبيقها على ما يزيد عن 10 مليون اسم ، وادت الى نتائج مرضية في وقت قصير نسبيا.

To my parents, brothers, sisters and my friends

Acknowledgements

I would like to express my gratitude to my advisor *Dr. Sami Taha* for his useful comments, suggestions and continuous support throughout the learning process of this master thesis. His guidance and knowledge have been a major factor in my ability to accomplish this work. He was very helpful and rich of ideas and solutions to overcome different challenges.

I would like to thank my thesis examiners, *Dr. Hashem Tamimi* and Dr. *Beverly Jamison* for their valuable comments and suggestions to improve the quality of this work. Also, I would like to thank the master program coordinator, *Dr. Liana Tamimi* for her efforts during my thesis registration and defense procedures.

I would like to thank American Psychological Association for granting us an access to their database and for providing us with the environment to conduct our experiments.

I would like to thank my friend *Mr. Ahmad Tamimi* for the technical support and continuous help on the way. I would also like to thank my work colleagues who were involved in the preparation of the reference datasets used for testing and validation of our algorithm. Without their passionate participation and input, the validation process could not have been successfully conducted. Also, I would like to thank my colleagues *Mrs. Khadija Qaraqe* and *Mr. Jumah Zwahera* for preparing the additional datasets used to test the generalization of the algorithm.

Finally, I should express my very profound gratitude to my parents and to my brothers, sisters, and friends for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment of this thesis would not have been possible without them.

Declaration

I declare that the Master Thesis entitled "A Heuristic-based Approach for Author Name Disambiguation" is my original work, and hereby certify that unless stated, all work contained within this thesis is my own independent research and has not been submitted for the award of any other degree at any institution, except where due acknowledgement is made in the text.

Contents

List of Figures

List of Tables

Acronyms

AD Author Disambiguation

LNFI Last Name First Initial

AC Author Contribution

AM Author Match

AP Author Profile

MT Matching Threshold

PSM Professor Students Model

TMM Transitive Matching Model

ORCID Open Researcher and Contributor ID

APA American Psychological Association

GTP Grond Truth Profile

OP Output Profile

CP Complete Profile

IP Incomplete Profile

CE Complete Profile with Error

IE Incomplete Profile with Error

Chapter 1

Introduction

In publication solutions, an author might write his/her name in different forms. This causes the machine to handle each name string as different author. In addition, there might be more than one author with the same name string. This issue leads to falsely merging the records of both authors. In this thesis, we address those issues and we present a decent solution to disambiguate the authors' names strings.

1.1 Problem Statement and Motivation

Researchers nowadays depend on digital libraries to keep up with the latest research in their field of interest. Usually, users of digital libraries, search for publications by author name and look for the contributions of a given person. When the number of publications in the dataset is large, the author name ambiguity problem is amplified, and name string alone is not sufficient to uniquely identify an individual author. Thus, to enhance the services of digital libraries, there is a need to distinguish authors with ambiguous names from each other.

Author disambiguation (AD) algorithm is mainly used to identify the real owner for a given scientific publication, and as a result, all publications of a given author can be aggregated in the same profile. Creating author profiles can be very useful for scientific communities in many different ways. First, the ownership of the contribution should be given to the right person, and thus, the assessment of authors using their contributions would be easier. Second, it can be used to create links to on-line resources such as author's home page and paper's full text.

In addition, authors' profiles can be used to create new resources such

as citation and collaboration networks. These networks can be a very useful source of information. For example it can be used to infer new information about authors such as identifying different communities, co-authorship information and the influence of authors on each other.

In the past, the disambiguation effort was performed manually by librarians. However, with massive publications and large digital libraries, manual disambiguation becomes very difficult and error-prone. Thus, many automatic disambiguation methods have been developed to deal with AD problem.

Mainly, the meta-data of a scientific publication such as co-authorship, affiliation, emails, etc. or even information extracted from the full text are used to determine the author identity. In this work, we propose a new algorithm that uses the publications' meta-data to perform the disambiguation task. Our algorithm is designed to be efficient in terms of performance and accuracy.

1.2 Main Challenges

Researchers face different challenges while addressing the author disambiguation problem. For example, the name "Jing Zhang" appeared on 54 papers in DBLP database, while it refers to 25 different individuals[35]. In addition, there have been three different students who share the same name *Yi Li* who have graduated from the same lab and share the same first co-author[35].

Name ambiguities problem appears due to many different reasons. According to Smalheiser and Torvik [34], author name disambiguation comprises four distinct challenges:

- A single individual may publish under multiple names:
  This happens due to many reasons, some of them are:
  a)name variants, b) spelling errors, c) name changes over time because of marriage, religious conversion and d) the use of nick names.
- Many different individuals have the same name:
  Fore example, common names are shared between many individuals.
- The incomplete or unavailable meta-data:

Table 1.1: Types of Name Variations[32]

Name variations	Examples
Abbreviations	L Zhang & Lei Zhang
Nick names	Chris Carter & Christine Carter
Middle names	L. S. Zhang & L Zhang
Hyphenated names	Jack-Thomas & Jack Thomas

For example, authors' middle names, their email addresses, or other information such as their affiliation are not available or incomplete.

- Multi-disciplinary and multi-institutional publications:
  Recently many articles are published as result of collaboration work between different institutions and/or as result of inter-disciplinary efforts.

One of the main sources of name ambiguities problem is the different name variations that can be used for the same name string. Sometimes, people hide their middle names/initials and use abbreviations or even nick names in place of their full name [32]. Table 1.1 shows different cases of name variations. This type of ambiguities is also known as synonym problem where different name variations refer to the same individual.

On the other hand, the homonym problem occurs when the same exact name string is shared between many different individuals. For instance, the name string *Collett, Thomas S.* that appears identical in four different articles refers back to four different individuals. In spite that this name string is exactly the same in these four different articles, it belongs to different persons.

In addition, AD problem becomes more challenging when an individual/author moves from one institution to another and thus changing his affiliation and may be his coauthors. The same situation occurs when the author changes his field of interest or changes his work group that may lead to changing his coauthors. Moreover, some authors are very active and they work with different collaboration groups or work for different institutions at the same time or move between them continuously.

Furthermore, insufficient and some times unavailable meta-data make the disambiguation task more difficult. For example, some individuals tend to work alone and thus the co-authorship information is not available. n addition, the dataset contains many papers written one or more centuries ago, long before computer processing of text data was envisioned or invented. These would not contain email addresses and many did not conform to current standards with respect to citation details such as affiliation or even clear identification of the publication and date.

Moreover, many papers especially the older ones are scanned using one of optical character recognition (OCR) technologies before ingestion. OCR scanning is vulnerable to errors and therefore the scanned records might contain some typos in names, affiliation, email addresses, etc.

1.3 Contribution

In the past, AD was resolved manually by Librarians. However, in the case of large scale digital libraries, manual disambiguation becomes very hard, time-consuming and error-prone. Therefore, automatic disambiguation methods are needed to resolve name ambiguity problems with minimum human intervention. In this thesis, we present a decent solution to AD problem using a heuristic-based algorithm.

Since our algorithm follows a heuristic approach, it does not require any training data. Preparing a reliable training data is difficult and time consuming task. This is because the training examples should be representative in terms of quantity and quality. In addition, there is a need to ensure the balance between negative and positive examples.

In our open world, we have a large amount of publications written by large number of authors. Processing millions of names and all their features might be inefficient due to the large number of comparisons. We need the process to be as efficient as possible and as accurate as possible. For significant number of name strings, a decision can be made by a subset of features. Therefore, we implement three stages of scoring and decision making so that not all processing and scoring have to take place. For higher efficiency, our algorithm is designed so that the computation tasks can be run in parallel.

To handle the wide variety of cases and name variations in a large database, the proposed algorithm is designed to be adaptive. Adaptivity means the

ability of the algorithm to change its behavior based on the handled case. For example, we consider the popularity of the name when taking the disambiguation decision so that we are more conservative when handling common names and more lenient with rare names. Another example is Chinese names which are more challenging than other names due to their high level of ambiguities, and thus, they need to be treated differently.

In order to tune and evaluate our algorithm, we prepare different reference datasets that cover a wide variety of names from different places and cultures. These datasets vary in size (small, medium and large), and encompass different challenges that help to ensure the generalization of the algorithm. Author names in these datasets are manually disambiguated by human efforts.

When the disambiguation algorithm is applied on a real large database, it becomes hard to evaluate the results of that algorithm on the entire database. In this work, we propose new techniques to assess the results of the algorithm when it is applied in a real environment using some quantitative measures. In spite that those measures might not exactly reflect the actual algorithm results, they can be used to estimate the accuracy of the results and ensure that the error rate is within the expected range.

1.4 Thesis Structure

The rest of this thesis is organized as follows. Chapter 2 introduces the previous work related to AD problem. In Chapter 3, we describe the algorithm design, data and methods. Then, we show the experiments and the results of our algorithm in Chapter 4. Finally, we present the conclusion and some suggested points for future work in Chapter 5.

Chapter 2

Literature Review

2.1 Introduction

In this large world with a population of about 7.4 billions of people [40], the name string is no longer a unique ID for an individual. Popular names such as *John Smith* are usually shared between many people. For example, in the US it was estimated that about 300 common names are shared between more than 114 millions of people [35]. Moreover, in China, just 129 surnames are shared between about 1.1 billion people [33]. Therefore, using the name string only, we will not be able to completely distinguish individuals.

The process of distinguishing the real owner of scientific work is mainly known as Author Disambiguation (AD). The AD problem has been investigated under different names according to the domain in which the problem appears[35]. AD is also known as entity resolution, web appearance disambiguation, name identification, object distinction/matching, duplication elimination/detection and record linkage[6, 35].

In AD, we deal with two main name ambiguities known as synonyms and homonyms. Synonym problem occurs when the same individual publishes different publications using different variations of his name[22]. On the other hand, homonym problem occurs when the same exact name belongs to different individuals[22]. These problems occur according to different reasons such as writing typos, hiding parts of names and the use of abbreviations or nicknames.

During our literature review, we explored many different automatic solutions that have been proposed to solve AD problem. In the next sections, we

show the main features and different approaches proposed to deal with AD problem.

2.2 Disambiguation Features

Researchers in literature have used different features to disambiguate author names. Table 2.1 shows some of the used features in the previous literature. These features fall into three main categories[14, 24]: explicit(basic) features, implicit(extracted) features, and additional(web information) features. Explicit features can be easily extracted from records metadata without any extra effort such as coauthors, email address and affiliation features. Explicit can further be classified as author-related features such as email and name, and article-related features such as article title, publication venue and keywords. Most of the proposed algorithms use some of first category (explicit features).

In certain cases, the name strings are very ambiguous such that the basic features are not enough and additional features are needed to perform the disambiguation task. These features are either collected from the web (web information), or extracted after processing the existing features (implicit features).

Implicit features do not exist directly in the article meta data and can be explored from other features such as exploring the self-citation or the reference overlap between articles. Bhattacharya and Getoor [5], Huang et al. [19] and Culotta et al. [9] used implicit features to enhance their results.

As an example of implicit features, Levin et al. [24] investigated different citation-based features such as self-citation. They showed that the self-citation feature is the most important citation feature that can be approximated without investigating the full network. The use of self-citation feature is proved to enhance the accuracy of results. Also, their results showed that the other citation features such as the full citation network can add a non-significant improvements at the expense of their extraction overhead if they are not already in the database.

In order to make the disambiguation decision easier, some research explore additional information. Mainly, their methods rely on collecting more information about authors from the web such as their home pages and CVs. These information can be helpful in very ambiguous cases. However, this

Table 2.1: Disambiguation Features

Explicit(Basic) Features		Additional Features	
Article Related Features	Author Related Features	Implicit Features	Web Information
Article Title	Author Name	Self-citation	Author Web Page
Article Abstract	Name Prefix	Reference Overlap	Author CV
Article Keywords	First/Middle Initials		
Coauthors	Email Address		
Journal Title	Affiliation		
Publication venue			
Publication Year			

advantage comes at the expense of the additional overhead needed to query the search engines to get this additional information. For example, to disambiguate authors in a citation record, Yang et al. [41] used the web correlation which represents the number of times the two citations co-occur in the web as a similarity measure.

Exploring external information leads to a trade off between the performance and the accuracy of the proposed algorithm. In other words, the gained accuracy obtained from investigating new features which may enhance the results comes at the expense of performance overhead of calculating these features and the complexity they add to the algorithm.

Some features such as email address, affiliation and coauthors are more reliable for AD than other features. For example, Momeni and Mayr [28] focused on the disambiguation of homonym names in the computer science bibliography DBLP using the co-authorship networks. Similarly, Kang et al. [22] depended mainly on the co-authorship information to discriminate authors with an assumption that co-authorship feature is clearer than other features. In addition, co-authorship feature is usually available and easily accessible as meta-data in scientific publications[22]. In spite of their promising results, their algorithm is evaluated against a data-set of Korean names only which are known to be generally clean. Korean names usually do not suffer from name variation problems. Korean name is usually written as a surname followed by a given name without delimiters or middle names[22]. Therefore, the results of this study, could not be generalized since it does not consider the variation of naming conventions in different cultures.

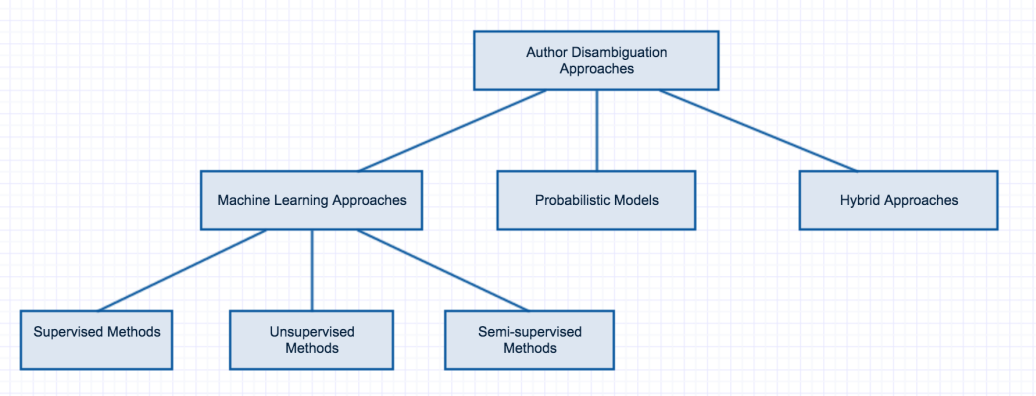


Figure 2.1: Disambiguation Approaches

2.3 Disambiguation Approaches

The AD problem has been investigated using several approaches in different domains ranging from individuals and researchers efforts to large projects in different institutions[11]. In this section, we discuss the different approaches that were proposed to deal with AD problem. Figure 2.1 depicts the main categories of AD approaches that have been investigated in the literature. In addition, Table 2.2 contains a summary of the characteristics of different methods that have been proposed to solve AD problem.

The main part of any AD method is to define a similarity function[14]. The input to this similarity function are the features of two publication such as (author name, coauthors, affiliation, email address, etc.) which usually represented as a feature vector. The similarity function finds the similarity between two features vectors extracted from two publications by calculating a similarity measure between the corresponding features and integrating them together to find the overall similarity score. The output of the similarity function is used to decide whether the two publications refer to the same individual or not based on a threshold.

The similarity measure between two corresponding features can be determined using different functions such as Levenshtein distance, Jaccard coefficient, Jaro, Jaro-Winkler, cosine similarity, soft-TFIDF, euclidean distance, etc.[14]. In other cases, a heuristic-based function is used, such as the number of terms or coauthor names in common, exact match such as email addresses or special values such as the last names with first/middle name initials[14].

One of the simplest methods to solve AD problem is the one that only depends on the author name strings. For example, Milojević [27] presented a simple initial-based algorithm that uses only the name string information to disambiguate authors. However, these methods are too simple to discriminate very ambiguous names in a large database that contain records of a large number of people from different cultures and backgrounds. However, these methods can be used to reduce the scope of candidate matches to given name string.

Most of the proposed algorithms use the last name first initial (LNFI) blocking method to cluster name strings that might belong to the same individual in the same cluster of candidates. Then, the comparisons are performed between elements inside the same LNFI cluster, instead of conducting all-against-all comparisons on the level of the entire database which is very expensive and inefficient in terms of performance and scalability.

2.3.1 Machine Learning Approaches

Many researchers make use of the machine learning approaches to disambiguate author names. Mainly, there are three machine learning methods namely, supervised, unsupervised and semi-supervised.

Supervised approaches rely on a machine learning classifier trained using a well-prepared training data set to find the matches of a given name string. The trained classifier (like random forest or support vector machines) is then used to label authors as matches or non-matches.There are different efforts that are based on the supervised approaches such as in [9, 17, 29, 37].

Under supervised methods, Han et al. [17] proposed two supervised learning approaches including a Naive Bayes based classifier and support vector machine(SVM)based classifier. Both methods depend on three citation attributes:co-author names, paper title, and journal title. The two approaches were evaluated using two datasets, one collected from the web using publication lists from homepages and the other collected from the DBLP citation database.

Treeratpituk and Giles [37] applied random forests classifier on random dataset consists of 6,803 articles from Medline database. These articles are chosen using the LNFI of 91 randomly selected author names. The author names in this dataset are manually labeled using the article metadata and

additional information(e.g. the Internet) whenever needed. The manually labeled dataset is used for both: training the classifier and evaluating the results. Their experiments showed that random forests classifier with an an accuracy of about 90% outperforms previous techniques such as Support Vector Machines (SVM).

The main challenge in supervised methods is preparing a representative and balanced training data that compose enough positive and negative examples with enough quantity to train the classifier especially for large digital libraries. Training data are either constructed manually by librarians or automatically using high precision features such as email and co-author features.

On the other hand, unsupervised methods such as in (Han et al. [16, 18], Bhattacharya and Getoor [5], Cota et al. [8]) require no training data. Alternatively, unsupervised algorithm define a similarity measure between two articles. However, in unsupervised methods, there is a need to develop a method to integrate the different similarity measures results from each feature into one similarity score[22].

Usually, unsupervised methods depend on a certain clustering algorithm to aggregate the publications of an individual into the same cluster that represents that real life author. For instance, Han et al. [16] applied k-means clustering algorithm by partitioning n records into k clusters in which each record belongs to the cluster with the nearest distance. Two years later, Han et al. [18] proposed a k-way spectral clustering method to address AD problem. The main challenge in these two methods is how to estimate the actual number of authors correctly. In addition, Huang et al. [19] used a DBSCAN[13] clustering method in order to cluster papers by author name.

Cota et al. [8] presented a heuristic-based hierarchical clustering (HHC) algorithm in bibliographic citation records. Their method combines different similarity functions with some heuristics to aggregate the information of author names in their proper clusters. They used a subset of the DBLP database in order to evaluate their work.

Arif et al. [4] proposed a multistage hierarchical method to deal with AD problem. Their method creates clusters in stages, for each new stage a new feature that differs from the previous stage is used. Their experiments on publications selected from DBLP achieved an average F1-score of about 92.33%.

In the area between the supervised and unsupervised methods, another approach called semi-supervised approach that tries to combine between the advantages of supervised and supervised methods. In semi-supervised approach or some times called weakly supervised approach, part of the algorithm is responsible for preparing some training examples to train the classifier.

Levin et al. [24] proposed a semi-supervised (self-supervised) algorithm to address AD problem in large scale databases. The algorithm consists of two stages. In the first stage, a high precision features such as email and co-authorship are used to collect a training set of positive and negative samples. In the second stage, the collected training data are used to train a features-based classifier which is, in turn, used to predict whether a given two articles refer to same author or not. The algorithm is applied and evaluated against Thomson Reuters of knowledge database which contains about 54,000,000 author instances with F1 of about 80.7%.

Training data can be prepared manually as in [17, 30] or collected automatically such as in [36] using a set of rules based on some important and reliable features such as the email and co-authorship information. Automatic generation of training data is vulnerable to errors and may lead to biased training data. For example, when the email address is used to prepare a set of positive samples(matches), the resulted training data could be biased toward new records[34].

The sufficient amount of needed training data depends on the complexity[34] and the size of the dataset to be disambiguated. Large datasets require more training data in order to be representative.

2.3.2 Probabilistic Models

Some researchers leverage the basic idea of probability theory to decide whether two name strings are matches or not. For instance, Torvik et al. [36] presented a probabilistic model for estimating the probability that two name strings in two different Medline articles belong to the same author[36]. Their method is based on Bayes' theorem shown in equation 2.3.1 below. It finds the probability that the compared record is Match(M) given that the similarity measure is S.

$$P(M \mid S) = \frac{P(S \mid M)\, P(M)}{P(S)} \tag{2.3.1}$$

Tang et al. [35] proposed a unified probabilistic model based on Hidden Markov Random Fields(HMRF). HMRF is derived from the classical Hidden Markov Model(HMM). HMRF represents features function with weights of this function determines the importance of each feature. The challenge in this method is the design of HMRF model which needs both estimating the weights and assigning the papers to their real authors.

The main challenge of probabilistic models is estimating the prior probability which needs some prior information about the entire database. For example, the Bayes' theorem needs the probability of matches which means percentage of similar authors in the dataset. However, this information is not usually available before performing the disambiguation task. Torvik et al. [36] proposed three different ways to estimate such probabilities.

2.3.3 Other Approaches

AD problem has been investigated using different other approaches such as incremental methods, citation-based methods in addition to a combination of different approaches.

One of the hardest problems in digital Libraries is the name ambiguities in the context of bibliographic citations[15]. On et al. [30] considered the problem of ambiguous author names in bibliographic citations. In addition, they study and compare different approaches to correctly identify the name variants that refer to same real world author in citation records[30]. Other examples of efforts in this area were presented in [30, 10].

2.3.3.1 Incremental Methods

AD methods also can be categorized into two classes according to the way in which the ingestion of new records into the database is managed. Methods that consider the ingestion process without the need to re-disambiguate the entire database are called incremental methods. Incremental methods such as in [10, 32, 37, 12], do not require reapplying the disambiguation algorithm against the entire database whenever new records ingested into the database.

In contrast, the non-incremental methods deal with a batch set of records in a pre-existing database. Most of the proposed methods are non-incremental methods that deal with the disambiguation effort as a batch process. Recently, researchers started to pay more attention to disambiguate authors'

names during the ingestion process of new records into the digital libraries.

For example, de Carvalho et al. [10] proposed an unsupervised incremental disambiguation method (INDi) that disambiguates names in each new ingested citation record. Their method mainly finds whether the new record belongs to one of the preexisting authors or not by calculating a similarity measure between the new record and its candidate matches. If no similar records are found for the new ingested record, it is considered to belong to new author.

de Carvalho et al. [10] assumed that the entire database is already disambiguated. Moreover, their algorithm was designed assuming a clean digital library, i.e. the preexisting author names are correctly clustered. In real cases the clean database assumption is not valid. To account for that, Esperidião et al. [12] proposed a method to aggregate incorrectly segregated author names clusters.

Chin et al. [6] proposed a geographical-based AD approach. Their approach won the first prize of the Track 2 of KDD Cup 2013 from Microsoft Academic Search competition which aims at determining duplicated authors in a data set[6]. The first step of their method considers distinguishing Chinese from non-Chines names using two dictionaries that cover different Chinese names. Their assumption that Chinese names have different characteristics(different naming conventions) and should be handled separately. Their method achieved about 0.99 F1-score.

Some incremental methods look at the disambiguation effort as consisting of two phases. First, the disambiguation is performed against the entire database. Then, in the second phase, a new method is designed to deal with loading new data into the database saving the time of repeating the whole process again. Under this framework, Qian et al. [32] proposed a comprehensive disambiguation algorithm that consists of two stages: the first stage is called BatchAD that disambiguate the entire database. Then, in the second stage which called IncAD, the disambiguation is done incrementally as new records ingested into the database.

2.3.3.2 Hybrid Methods

Other AD solutions use a combination of two or more approaches(hybrid approach) to address AD problem. For instance, Han et al. [16] used a

K-mean clustering algorithm(unsupervised method) based on an extensible Naive Bayesian probability model (Probabilistic Model) used to compute the distance between the citation and the cluster in order to disambiguate names in citation records. In this model, only three features (paper title, journal title and coauthors) are used.

Another example of hybrid approach is the work done by Huang et al. [19] who suggested two steps machine learning approach to solve AD problem. In the first step, a blocking method based on the non-conflicting name variations to retrieve candidate classes of authors with similar names. Then they used a DBSCAN[13] clustering method in order to cluster papers by author. The distance metric between papers used in DBSCAN is calculated by an online active selection support vector machine algorithm (LASVM) that intended to be faster with lower testing error than standard SVM[19]. In addition, Nguyen and Cao [29] proposed a hybrid method consisting of two phases to detect names in text and link them to proper entity in Wikipedia.

In this thesis, we present a heuristic-based approach to address AD problem. Unlike supervised methods, our approach does not require training data. Also, our heuristic approach is more flexible to model real cases and accounting for causes of errors. The proposed algorithm allows for better controlling of the clustering process to ensure the accuracy as well as the efficiency of the algorithm.

In addition, our algorithm is designed for batch processing of author names (non-incremental approach). However, the core scoring methods can be used to design an incremental solution for newly ingested data.

Table 2.2: Summary of Disambiguation Methods

Ref.	Approach	Technique	Database	Evaluation Measures	Incremental	Features
[19]	Supervised method	Online Active Support Vector Machine	CiteSeer	precision, recall and pairwise F1	No	affiliation, email, venue, topic
[36]	Probabilistic Model	Bayesian probability	MedLine	precision, recall, and accuracy	No	title, name attributes, journal name, coauthors, affiliation, medical subject headings
[35]	Probabilistic Model	Markov Random Fields	Arnetminer.org	pairwise precision, recall and F1	No	title, publication venue, year, abstract, coauthors, references
[27]	unsupervised (name string only)	First Initial, All Initials similarity measures	simulated dataset (set of articles selected from different domains)	Accuracy	No	last name, first initial, middle initial
[30]	Unsupervised	-String-based distance (Jaccard, Jaro, Jaro-Winkler, TFIDF) -Vecotor-based Cosine Distance	Selected data from 4 different domains (DBLP, e-Print, BioMed, EconPapers)	Accuracy	No	coauthors
[32]	Hybrid	Hierarchical agglomerative clustering for initial disambiguation and Probabilistic model for new records	CaseStudy and DBLP	B-cubed, precision, recall, F1	Yes	author name, title, abstract, publication venue, year, reference, keywords, coauthors,
[10]	unsupervised	Heuristic-based Hierarchical Clustering (HHC)	BDBComp and synthetic(unreal) data generated by SyGAR	K metric [23]	Yes	title, publication venue, coauthors
[9]	supervised	Error-driven Training Algorithm	DBLP, Rexa and Penn data	B-cubed and Pairwise (Preision, Recall and F1)	No	title, venue, email, coauthors, affiliation
[37]	supervised	Random Forests	Medline	Accuracy	No	title, affiliation, authors, publication date, journal, MeshHeading
[17]	supervised	Naive Bayes and Support Vector Machine	DBLP and data collected from web	Accuracy	No	title, journal, affiliation, coauthor
[16]	Hybrid (unsupervised+ probabilistic model)	K-means and Naiv Bayes	selected dataset for two names : (J Anderson and J Smith)	Accuracy	No	title, journal, acoauthors
[24]	semi-supervised	L1 Regularized Logistic Regression,classifier	Thomson Reuters Web of Knowledge	B-cubed, F1	No	title, abstract, keywords, language, name attributes, affiliation, year, citation features, conjunctive features

Chapter 3

Data and Methods

In this chapter, we describe the design principles of the algorithm, the overall algorithm methodology in addition to the details of different scoring methods.

3.1 Definitions

- Synonym issue: the issue of having multiple name strings for one author.
- Homonym issue: the issue of having one name string for multiple authors.
- Author ID
  A unique identifier of the author(individual) that consists of a combination of author name string and the article id where the name string appears. Author ID is needed in order to distinguish authors who share the exact name (homonym case).
- Author contribution (AC)
  A record that contains all the information (title, coauthors, affiliation, etc) about an author (individual) with a given Author ID collected from the article that contains that author name string.
- LNFI
  Last Name First Initial (e.g. *Smith, A*)
- LNFI Cluster: A set of author contributions that share the same LNFI. For example, author contributions with names (*Smith Adam*, *Smith Ali* and *Smith Ahmad*) belong to the same LNFI cluster *Smith A*.

- AC_s vs. AC_c
  When comparing two ACs to find if they refer to the same person, the first AC is called Source AC (AC_s), and the record to which it is compared is called Candidate AC (AC_c). Authors who belong to the same LNFI cluster are considered candidates to each other.

- Matches vs. Non-matches
  Two or more ACs that refer to the same individual are called matches. In contrast, two ACs that refer to different individuals are called non-matches.

- MT
  Matching Threshold. AC_s and AC_c are considered as matches if their similarity score is equal or above MT.

- AM
  Author Match Record. For each AC record, there is a corresponding AM record that contains the result of comparing a given AC to all other ACs in the same LNFI cluster. The AM record for a given AC contains all its *matches*.

- AP
  Author Profile. This record contains the contributions that belong to a given author(person). The final output of the disambiguation algorithm is the collection of AP records.

3.2 Ground Truth Datasets and Labels

In order to evaluate our algorithm, we need a reference ground truth dataset. The reference dataset should represent a wide variety of real name string cases. This section describes the criteria and the process followed to build our ground truth data for validation.

The source of our reference datasets is the American Psychological Association (APA). APA is the leading scientific and professional organization representing psychology in the United States[1]. APA maintains a number of databases such as PsycINFO, PsycARTICLES and PsycBOOKS[2]. In particular, we collected our reference datasets from PsycINFO database.

PsycINFO is one of the major APA databases that contains publications from the 17th century to the present[3]. Its main focus is on the field of

psychology and the behavioral and social sciences in addition to some interdisciplinary related content[3].

3.2.1 Datasets Selection

Our criteria in selecting the datasets are oriented to have a variety of cases that would represent as many cases as possible in the entire database. Several datasets were selected. We followed the following steps in the selection process:

- Last Name is used as a key filtering approach. We retrieved from all author names of the selected last name and all their co-author name strings. Then, All documents that contain one of those author names are retrieved from APA Psychinfo database.

- Select several last names that might have different commonality/rarity to have a better representation of reality and different sizes of datasets (small, medium, large). This criterion enables us to evaluate the disambiguation for rare names and common names.

- Select last names from different cultural backgrounds (Mediterranean, Oriental, US...).

The *Selective* dataset represents an extreme case of author name strings. It contains 4456 author name strings (including homonyms). We tried to collect names from multiple backgrounds and common last names. The data is collected based on three different name filters: *Brown W, Xue H and Ramos M.*

3.2.2 Datasets and Labels

For the experimentation and evaluation, we used different ground truth datasets. These reference datasets are described in Table 3.1 below compiles the used datasets. Moreover, we used three additional datasets which do not participate in the algorithm tuning. These datasets shown in Table 3.2 are used to validate the generalization of the algorithm.

Authors in those datasets are manually disambiguated. To do this, all name strings that refer to individual author in each dataset is manually given a unique ID (label).

[1]This dataset is collected using 3 different LNFI names: "Brown, W", "Ramos, M"

Table 3.1: Ground Truth Datasets

Dataset	Query Criteria(Last Name)	Size(# of documents)	AC Records
Abbas	Abbas	382	1232
Collet	Collet	623	1529
Mahmood	Mahmood	276	912
Racine	Racine	335	752
Royer	Royer	331	764
Vail	Vail	655	1393
Zang	Zang	675	3539
Selective	Selective[1]	1221	4466

Table 3.2: Additional Datasets

Dataset	Query Criteria(LNFI)	Size(# of documents)	AC Records
PetersE	Peteres, E	291	1311
PaulS	Paul, S	417	2486
ZhangM	Zhang, M	898	5807

3.3 Evaluation Criteria

In this section we define the criteria and the measures we used to evaluate the results of our algorithm against the reference datasets.

3.3.1 Definitions

We compared the output profiles (OP) generated using our algorithm to ground truth profiles (GTP) in the reference datasets. Any OP profile can be classified into different types when compared to their corresponding GTP profiles.

1. Complete Profile (CP)
   OP exactly matches its corresponding GTP.

2. Incomplete Profiles (IP)
   Incomplete profile results when OP partially matches its corresponding GTP. One ore more contributions are missed from the OP but it

and "Xue, H"

contains no wrong contributions. This case occurs when GTP is split into two or more OP profiles.

3. Error Profiles (EP)
   In this case, contributions for different authors are merged together in the same OP. EP can be further classified into:

   (1) Complete Profiles with Error (CE)
   This type is similar to CP but with additional wrong contributions. It contains some outliers(one or more contributions for different author(s)).

   (2) Incomplete Profiles with Error (IE)
   This type is similar to IP but with additional wrong contributions. This is the worst type of error where some contributions are missed and extra wrong contributions are added to the same profile.

Figure 3.1 below shows the different possible types of algorithm output profiles for a given reference profile (GTP) that contains four different contributions (AC1, AC2, AC3 and AC4).

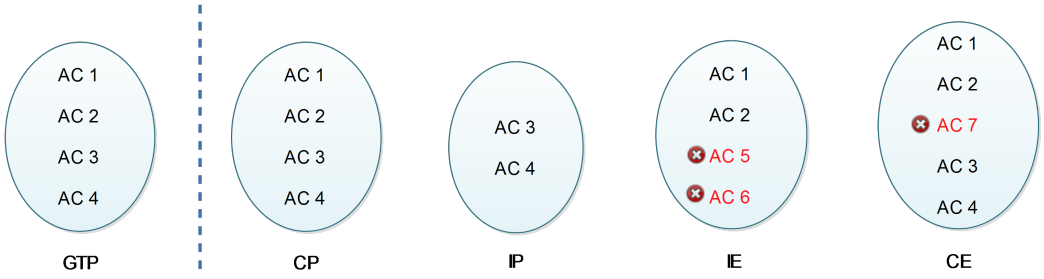


Figure 3.1: Different possible OP types for a given GTP

3.3.2 Evaluation Measures

Assume that $O = \{OP_1, OP_2, ...\}$ is the set of OP profiles generated by the algorithm and $GT = \{GP_1, GP_2, ...\}$ is the set of GT profiles in the reference dataset, then we use the following measures to validate our algorithm results:

1. CPR : Complete Profiles Rate

$$CPR = \frac{|CP|}{|O|}100\% \tag{3.3.1}$$

2. IPR : Incomplete Profiles Rate

$$IPR = \frac{|IP|}{|O|}100\% \tag{3.3.2}$$

3. EPR : Error Profiles Rate

$$EPR = CER + IER \tag{3.3.3}$$

Where,

(1) CER : Complete Profiles with Error Rate

$$CER = \frac{|CE|}{|O|}100\% \tag{3.3.4}$$

(2) IER : Incomplete Profiles with Error Rate

$$IER = \frac{|IE|}{|O|}100\% \tag{3.3.5}$$

According to these measures we denote the following:

- Our main goal is to maximize the CPR while minimizing the remaining rates. If our algorithm is ideal, then CPR will be 100% while other rates would be zeros.

- IPR represents the portion of the output profiles that have been divided into two or more profiles. IP profiles are not perfect as CP profiles, where contributions of same author aggregated in the same profile, but they are not as bad as EP profiles where contributions of different authors are falsely merged together.

- If our algorithm is lenient and inclined towards aggregation, then some IP profiles might become complete profiles and some other IP profiles might become error profiles. Thus, we should keep the balance between turning IP into CP while minimizing EPR.

- If the algorithm is conservative and inclined towards separation, then IPR rate will increase and we might reach a point where each profile contains only one contribution (singleton profiles).

3.4 Design Principles

During the design of our algorithm, we tried to model different real cases in the world of publications that would enhance the accuracy of the results and increase the author's satisfaction. In this section, we explain these main design principles and models.

3.4.1 Adaptivity

There is a wide variety of authors who come from different cultures and different backgrounds. This diversity leads to different naming conventions and different naming commonality/rarity. Our algorithm is designed to be adaptive based on the case being handled. In the following subsections, we discuss different adaptivity features supported in our algorithm.

3.4.1.1 Sliding Matching Threshold (SMT)

Author names with initials are a common source of disambiguation problems, especially in large LNFI clusters. In fact, we might find many authors who claim the ownership of a given AC with name string *Smith, A* since all ACs in this cluster share the same LNFI and are candidates to match this AC. For instance, ACs with names (*Lee, Jun*, *Lee, Juan*", *Lee, Jung*), all, compete to own any AC with name *Lee, J.* Thus, when the comparing ACs that contain initials we use the sliding threshold technique which slides the matching threshold between 0.4 to 0.6 using a variable penalty of about (0 to 0.2) based on the LNFI frequency (commonality of the name).

The size of LNFI cluster defines the competition rate between different ACs to own initial-name ACs. For example, if the size of LNFI cluster is 5000, then, there will be about 5000 ACs who are allowed to claim owning LNFI ACs. To account for this problem, we need more evidence from each candidate to assure that this AC really belongs to him. The larger the LNFI cluster, the higher the matching threshold is. In other words, for popular LNFI names (more competition), we need more evidence before assigning the contribution to the candidate owners. Thus, we increased the matching threshold as a function of name frequency (LNFI cluster size) according to the equation 3.4.1.

$$MT = MT_{min} + 0.2K/\mu \tag{3.4.1}$$

where

- MT_{min} : Minimum matching threshold. MT_{min} is set to 0.4.
- K : LNFI cluster size (LNFI frequency)
- μ : Maximum LNFI cluster size (maximum LNFI frequency)

3.4.1.2 Chinese and Short names

During our initial experiments and investigations, we observed that Chinese names are more challenging than other names because they are very ambiguous. First, there are many people who share the same exact name (the homonym case). Second, names are very similar to each other and there are many cases where the difference is only one or two characters, so if we want to account for typos in names, we might end up with falsely merging two contributions of totally different individuals. The same issue happens when the name is too short (e.g. Li vs Le).

To account for these cases, we adapt our algorithm to be more conservative for both Chinese and short names. To handle Chinese names, we used a list of very popular Chinese last names. Whenever a contribution has a last name that belongs to this list, we decreased the string score share for that name to 96% of the original score. This means that when comparing two contributions and one of them belongs to a Chinese common last name, the string score needs to be higher than normal in order to pass to the silver stage, otherwise, they two contributions will be marked to belong to different persons.

A similar strategy is followed in case of short first names. If the two compared first names are complete (no initials) and their length is less than or equal to three characters, then they should be exactly the same, otherwise they will lose 0.5 of the original string score and thus will not pass into silver stage. In other words, when the two compared first names are two-character or three-character length, they should be exactly the same and no typos are allowed.

3.4.1.3 Missing Feature

One of the issues that we face when comparing two ACs, is the absence of one or more features. If we want to be conservative, then when a feature is missed, then the similarity score will lose the weight of this feature. This might deprive the two contributions of being aggregated in the same profile just because one feature is missed for any reason. To account for this issue, we

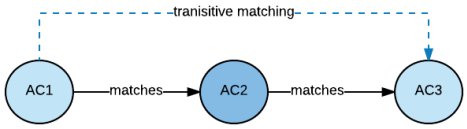


Figure 3.2: Transitive Matching

applied a technique that will give the opportunity for these two contributions to be merged even if one feature is absent. This is done, by transferring half of the weight given to the absent feature to the weight of the available feature.

For example, if the affiliation is not available, then half of the affiliation weight is given to the coauthor feature. In this way, the compared contributions have the chance to be merged if they achieve a relatively high co-authors similarity score (higher than it would be if the affiliation feature was available). By transferring part of the weight of the missed feature to the available one, we keep the balance between penalizing the similarity score for that loss of information and giving the chance to aggregate the compared contributions when the available feature shows a high similarity score.

3.4.2 Transitive Matching Model (TMM)

Transitive matching is one of the important design principles in the algorithm that helps to model changes in AD problem. TMM states that if a contribution AC_1 matches another contribution AC_2 and AC_2 matches a third contribution AC_3 then AC_1 matches AC_3. In other words, the three contributions AC_1, AC_2 and AC_3 will be aggregated in the same author profile. Transitive matching is depicted in Figure 3.2.

TMM is important to model interest changes and/or author movements. The assumption is that when the author changes his interest or affiliation, he passes through a transition state where he/she shares some information (coauthors/affiliation) with the old state.

TMM makes use of transition state to aggregate contributions of authors from different author's states. In spite of its clear benefits, TMM has some side effects that might lead to falsely aggregating different contributions of different persons in the same profile. An example of this side-effect is shown in Figure 3.3.

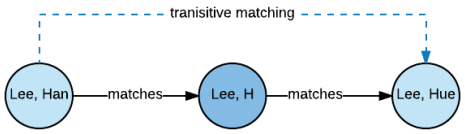


Figure 3.3: Transitivity Side Effect

3.4.3 Professor Students Model (PSM)

In academia, the same professor supervises different students. As a result, the professor usually publishes different articles with each student. For example, when the professor publishes an article with the first student(std1), we have two ACs; one for the professor AC_{1-prof} and the second for the student AC_{1-std1}. Similarly, when a new student(std2) publishes a new article with the same professor, we get new two author contributions; AC_{2-prof} and AC_{2-std2}.

If we look at this scenario from professor perspective, we know that AC_{1-prof} and AC_{2-prof} refers to the same person even though that the two ACs have different coauthors(std1 in AC_{1-prof} and std2 in AC_{2-prof}). However, the affiliation is shared between the two contribution. In this case, the affiliation feature should be enough evidence to aggregate the two ACs in the same profile (the professor profile).

On the other hand, from students' perspectives, we want AC_{1-std1} and AC_{2-std2} to be marked as non-matches even though that they share the same coauthor (their supervisor/professor) and they have the same affiliation. However, if the names of the two students are very similar, the algorithm might decide to falsely merge their contributions in one profile. This case becomes harder to detect when we have a homonym case where the two students share the same exact name.

3.5 Scoring Methods

We used different scoring methods to calculate the similarity score between two contributions. In this section, we describe the main scoring methods.

3.5.1 String Scoring

Our string score is mainly based on **Jaro distance metric** [7, 20, 21]. The string score algorithm is designed to account for different cases. The string score is calculated according to equation 3.5.1.

$$SS = (J_{sj} + I_{si})P_f P_c \tag{3.5.1}$$

where

- SS : string score
- $_j$: Jaro weight
- I_s : initials score
- $_i$: initials weight
- P_f : short first name penalty
- P_c : Chinese name penalty
- P_c : Chinese name penalty

using equation 3.5.1, we denote the following points regarding string score calculation:

1. Initials-aware string scoring:
   Classical editorial distance scoring methods such as **Jaro distance metric** might not behave as intended in some cases where names contain initials (first initial and middle initial). Thus, we added names' initials matching logic to equation 3.5.1. For example, names with different middle initials usually belong to different individuals.

2. Short first name penalty:
   A penalty is imposed on short first names (with length 2 or 3 chars). Complete first names with lengths 2 or 3 should be identical, otherwise, a penalty is imposed to reduce the score. This is needed since short names get a high **Jaro score** even if they refer to different persons.

3. Chinese name penalty:
   Chinese names are challenging especially with common last names and short first names. To account for that, a set of common Chinese last names (ccl) is used. When a last name belongs to *ccl*, the complete first names should be almost(0.96) identical.

4. Name consistency:
   We believe that all name variants for a given name should have the opportunity to be compared. However, in some cases, two name variations might get a string score that is below the string score threshold. Two names are considered consistent if one of them represents a variation for the other. To account for these cases, we added a name consistency checker to find candidates undetected by string score. Name consistency checker is enabled when the string score is below 90 and greater than or equal 70.

3.5.2 Co-authors Scoring

Let's assume that we have two co-authors lists to be compared: $C_1 = \{c_1, c_2, c_3, ...\}$ and $C_2 = \{c_1, c_2, c_3, ...\}$, then the co-authors score is calculated as shown in equation 3.5.4:

$$\zeta_f = C_1 \cap C_2 \tag{3.5.2}$$

Where ζ_f is the number of full names shared between two co-author lists.

$$\zeta_{lnfi} = (lnfi(C_1) \cap lnfi(C_2)) - lnfi(\zeta_f) \tag{3.5.3}$$

Where ζ_{lnfi} is the number of lnfi names shared between two co-author lists and lnfi(C) is a function that converts a list of full names (C) into a list of lnfi names.

$$CS = \zeta_{ff} + \zeta_{lnfilnfi} \tag{3.5.4}$$

where

- CS : coauthor score
- $_f$: ζ_f weight
- $_{lnfi}$: ζ_{lnfi} weight

$$CS_n = \begin{cases} 1, & \text{if } CS \geq 1 \\ CS, & \text{Otherwise} \end{cases} \tag{3.5.5}$$

Where CS_n is the normalized coauthor score.

Generally, coauthor list is not clean and coauthor name is written with initials instead of using the full name. To account for these cases where the full names are abbreviated or misspelled, the lnfi matching measure (equation 3.5.3) is added to the scoring equation but with lower weight than the full name matches.

3.5.3 Affiliation Scoring

From one publication to another, the affiliation of the author might be written in different ways. For example, the author might add some new words, remove other words, or he might use some abbreviations. Thus, the affiliation needs to be normalized before it can be used in the scoring. For example, common words such as (department, university, center, ...) or their equivalent abbreviations such as (Dept., Univ, Centr) are removed.

The output of the affiliation normalization process is a set of normalized items that represent the important parts of the affiliation. Given two normalized affiliation lists: $aff_1\{item1, item2, ...\}$ and $aff_2\{item1, item2, ...\}$, the affiliation score (AS) is calculated according to the following equations.

$$\chi = min(|aff_1|, |aff_2|) \tag{3.5.6}$$

$$AS = \begin{cases} 0, & \text{if } \chi{=}0 \\ (aff_1 \cap aff_2)/3, & \text{if } 0 < \chi < 3 \\ (aff_1 \cap aff_2)/\chi, & \text{Otherwise} \end{cases} \tag{3.5.7}$$

We used the minimum as normalization for the affiliation score in order to account for the cases when the author adds new words to the same affiliation. Since the affiliation is normalized and contains in most cases the most important words in the affiliation, it is safe to use the minimum for normalization.

Table 3.3: Features Weights Distribution

Feature	Weight
String Score	0.1
Affiliation Score	0.4
Co-author Score	0.5

3.5.4 Weights Distribution

The decision in the silver stage is taken based on three main features, string score, coauthors and affiliation. The distribution of weights among these features is shown in Table 3.3. We used a low weight (0.1) for string score for two reasons. First, it is already used in the golden stage as a key to enter the silver stage. Second, we need to reduce the impact of string score in homonym cases where string score might reach 100%. Thus, we don't need the string score to be dominant which might lead to false positives in the case of very ambiguous and similar names.

Initially, we distributed the remaining weight (0.9) equally between the two remaining features (affiliation and co-author). However, our initial experiments showed that the coauthor feature is more reliable than affiliation since the affiliation can be shared among a large number of persons, especially in large-scale universities and institutions. Thus, we set the affiliation weight to 0.4 and the co-author weight to 0.5. In addition, according to [39], the domain experts gave higher weight for co-author feature than the affiliation feature.

3.6 Algorithm Design

Our algorithm consists of three phases pipeline. The first phase prepares the input to the next phase including features extraction and data pre-processing. In the second phase, the matching logic is performed and the similarity score is calculated to decide whether any given two name strings belong to the same individual or not. Finally, in the third phase, the output of the second phase is used to aggregate similar records of an individual in the same profile. The overall algorithm design and workflow is shown in Figure 3.4.

To ensure high accuracy results, we used the most important features among the article's meta data. Mainly, we used email, name string similar-

ity, affiliation and co-authors features. In addition, we involved the publication date and frequency of the name in the disambiguation decision. We neglected other features such as title and keywords because our initial experiments showed that these features have an insignificant impact on the result's accuracy, while at the same time, they slow down the algorithm because of extra features calculations.

In addition, our heuristic approach includes both aggregation and separation techniques. For example, the algorithm might take a positive decision (the two publications belong to the same individual) using only one feature with strong evidence (high score), or using a combination of features where each of them plays a role to strengthen the similarity score. For example, if the two publications share two or more full name co-authors, then it is more likely that these publications refer to the same individual.

On the other hand, we include some techniques to take a negative decision (the two publications belong to different individuals). For example, if the difference in publication date is more than the maximum expected publication life, it is more likely that the two publications refer to two different individuals.

3.6.1 Contribution Extractor (CE)

Contribution Extractor (CE) is responsible for collecting all information of a given contribution in the same record called Author Contribution (AC). In addition, CE extracts and normalizes the features such as coauthor and affiliation list so they become ready for comparison in the next component. CE accepts a set of original records and outputs a set of AC records. CE is designed to work in parallel to speed up the process. Figure 3.5 depicts an input/output example of CE component.

3.6.1.1 Affiliation Normalization

The affiliation normalization includes the following items

- Remove (nullify) street addresses starting with a number
- Remove common affiliation words such as Institution, School, Hospital, Center, Department, Dept, etc.
- Remove stop words such as (is, the, or, and, etc.).

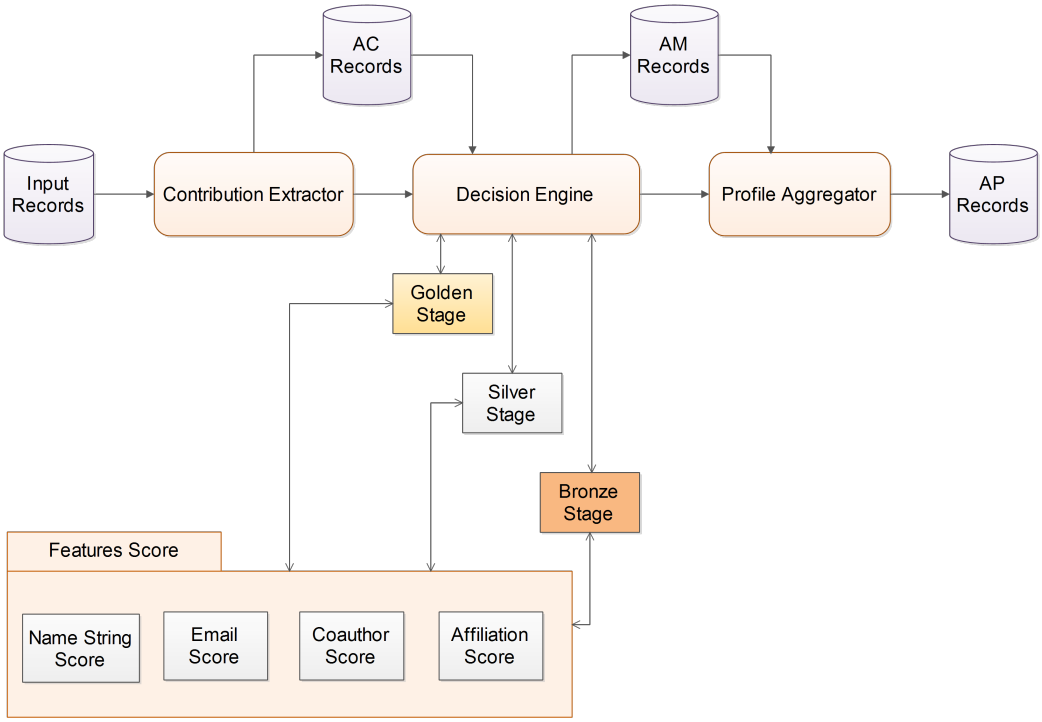


Figure 3.4: Algorithm Workflow

3.6.2 Decision Engine (DE)

Decision Engine (DE) represents the brain of the algorithm. In this phase, the matching decision is made. Since DE is the core component that contains lots of computations, it is the slowest component in the algorithm. To speed up this component, the computation is decomposed into three stages: Golden stage, Silver stage and Bronze stage. Algorithm 1 describes the DE logic.

Each stage builds on the result of the previous stage and returns ternary number to represent three statuses, match, non-match, and un-decidable. If a *match* or a *non-match* is returned, then the decision is made, and no need to go to the next stage. If un-decidable is returned, the next stage will be used to attempt to come up with a decision. Our goal is to take the matching decision as soon as possible with the minimum computations. The lines 10 and 16 in Algorithm 2 represent the transition state between stages.

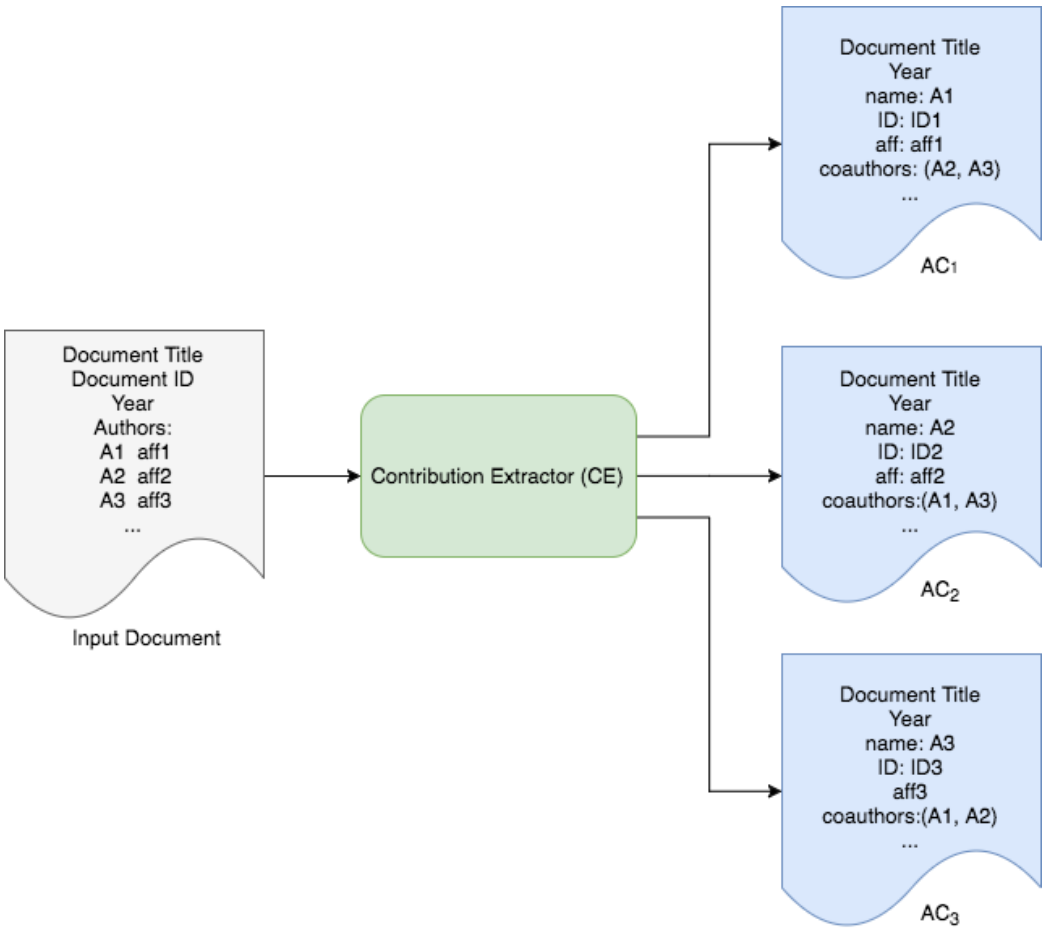


Figure 3.5: Contribution Extractor (CE)

Result: AM record for each input AC record. AM contains all matches for a given AC

Function *DecisionEngine(AC* $\{AC_1, AC_2, ...\}$*)*
  foreach *AC record in AC* $\{AC_1, AC_2, ...\}$ *Records* **do**
    FindMatches(AC)
  end

Algorithm 1: Decision Engine Algorithm

```
Result: AM_s: a record that contains a set of AC_c records that are
        similar to AC_s
1  Function FindMatches(AC_s)
2      AM_s ← {}
3      foreach candidate AC_c in lnfi(AC_s) do
4          isMatch ← GoldenStage(AC_s, AC_c)
5          if isMatch = match then
6              AM_s ← (AM_s ∪ AC_c)
7          else if isMatch = nonMatch then
8              get next AC_c
9          else
                                          /* go to Silver stage */
10             isMatch ← SilverStage(AC_s, AC_c)
11             if isMatch = match then
12                 AM_s ← (AM_s ∪ AC_c)
13             else if isMatch = nonMatch then
14                 get next AC_c
15             else
                                          /* go to Bronze stage */
16                 isMatch ← BronzeStage(AC_s, AC_c)
17                 if isMatch = match then
18                     AM_s ← (AM_s ∪ AC_c)
19                 else
20                     get next AC_c
21                 end
22             end
23         end
24     end
```

Algorithm 2: Find Matches Algorithm

3.6.2.1 Golden Stage

The golden stage is mainly based on email feature. Even though the email sounds like a feature that can be used alone to disambiguate people, but there are few issues in the data that makes it not completely reliable. Below is a list of the issues that degrade the significance of the email:

- Not all author names in the database are associated with email. In fact, the majority of authors' name strings are not associated with emails.
- Some authors use different emails in different articles. This issue makes it impossible to use the email feature alone in disambiguation.
- Different authors might use same emails in different publication. For example, there are cases where two different authors from the same department use same department's email on their publications.

To tackle these issues with email, we combine the email feature with LNFI as a matching criterion. In other words, if two contributions have the same email and they share the same last name and first initial, then they are considered as matches.

On the other hand, if any of the following rejection criteria holds, then the two contributions are considered as non-matches:

- ACs that are generated from the same article are considered as non-match. Authors who coauthored the same article are usually different individuals even though they share the same LNFI.
- ACs with more than 80 years publication date difference (the maximum expected publication period), usually belong to different individuals.

If neither the matching criteria nor the rejection criteria hold, then the case is considered as un-decidable and it is passed to the next stage (silver stage). The golden stage is described in Algorithm 3.

3.6.2.2 Silver Stage

since co-authors and affiliation features are the most important features after the email, the silver stage is mainly composed of a combination of co-author scoring and affiliation scoring. In addition, this stage adds some adaptivity features to the algorithm such as considering name commonality when taking the disambiguation decision. The logic of silver stage is described in Algorithm 4.

Result: returns if two names refers to same author or not

Function *GoldenStage(AC_s,AC_c)*
  if $email_s = email_c$ **then**
    return *"match"*
  else if $|PubYear_s - PubYear_c| \geq 80$ **then**
    return *"nonMatch"*
  else
    go to next stage
  end

Algorithm 3: Golden Stage

Result: returns if two names refers to same author or not

Function *SilverStage(AC_s,AC_c)*
  $isMatch \leftarrow false$
  $minMatchingTh \leftarrow 40$
  $nextStageTh \leftarrow 20$
  $nameScoreTh \leftarrow 90$
  $SS \leftarrow$ calculate string score$(name_s, name_c)$
  if $SS < nameScoreTh$ **then**
    return *"nonMatch"*
  else
    $AS \leftarrow$ calculate affiliation score(aff_s, aff_c)
    $CS \leftarrow$ calculate coauthor score(co_s, co_c)
    $SS = SS - nameScoreTh$

$$TotalScore = SS_n + CS_c + AS_a \tag{3.6.1}$$

    $MatchingTh = minMatchingTh + \delta$
    if $TotalScore \geq MatchingTh$ **then**
      return *"match"*
    else if $TotalScore \geq nextStageTh$ **then**
      go to Bronze stage `/* go to next stage */`
    else
      return *"nonMatch"*
  end

Algorithm 4: Silver Stage

3.6.2.3 Bronze Stage

The bronze stage tries to find matches with low Silver stage score but with low-frequency names. The main purpose of this stage is to decrease the number of incomplete profiles by aggregating profiles with rare names that have low similarity score. The assumption here is that contributions with rare names (low-frequency names) are more likely refer to the same individual if there is similarity match between them even though that their similarity score is low. We consider a given name as a rare name if it is frequency is less than or equal *10*. This stage is described in Algorithm5.

In order to reduce the possibility of falsely merging two profiles of different persons, we excluded names with initials only and thus only full names might pass this stage. In addition, the string score between the compared names must be *100%* (exact name match). Moreover, this stage requires a Silver stage similarity score to be more than or equal *20%*.

Result: returns if two names refers to same author or not
Function *BronzeStage(*AC_s,AC_c, *nameScore)*
  $f_s = freq(name_s)$
  $f_c = freq(name_c)$
  if *(nameScore* $= 100$ ***AND*** $f_s \leq 10$ ***AND*** $f_c \leq 10$*)* **then**
    return *"match"*
  else
    return *"nonMatch"*
  end

Algorithm 5: Bronze Stage

3.6.3 Profile Aggregator (PA)

This component is responsible for aggregating all contributions of the same author in the same AP record. It accepts AM records as input and outputs AP records. Similar to the previous components, PA is designed to work in parallel. This component makes use of the transitivity feature to aggregate the contributions of the same individual in the same profile. Figure 3.6 depicts an example of the transitive feature of the profile aggregator component.

This component aggregates all contributions of a given author in one record called author profile (AP). AP record is given a unique ID that belongs to a single individual.

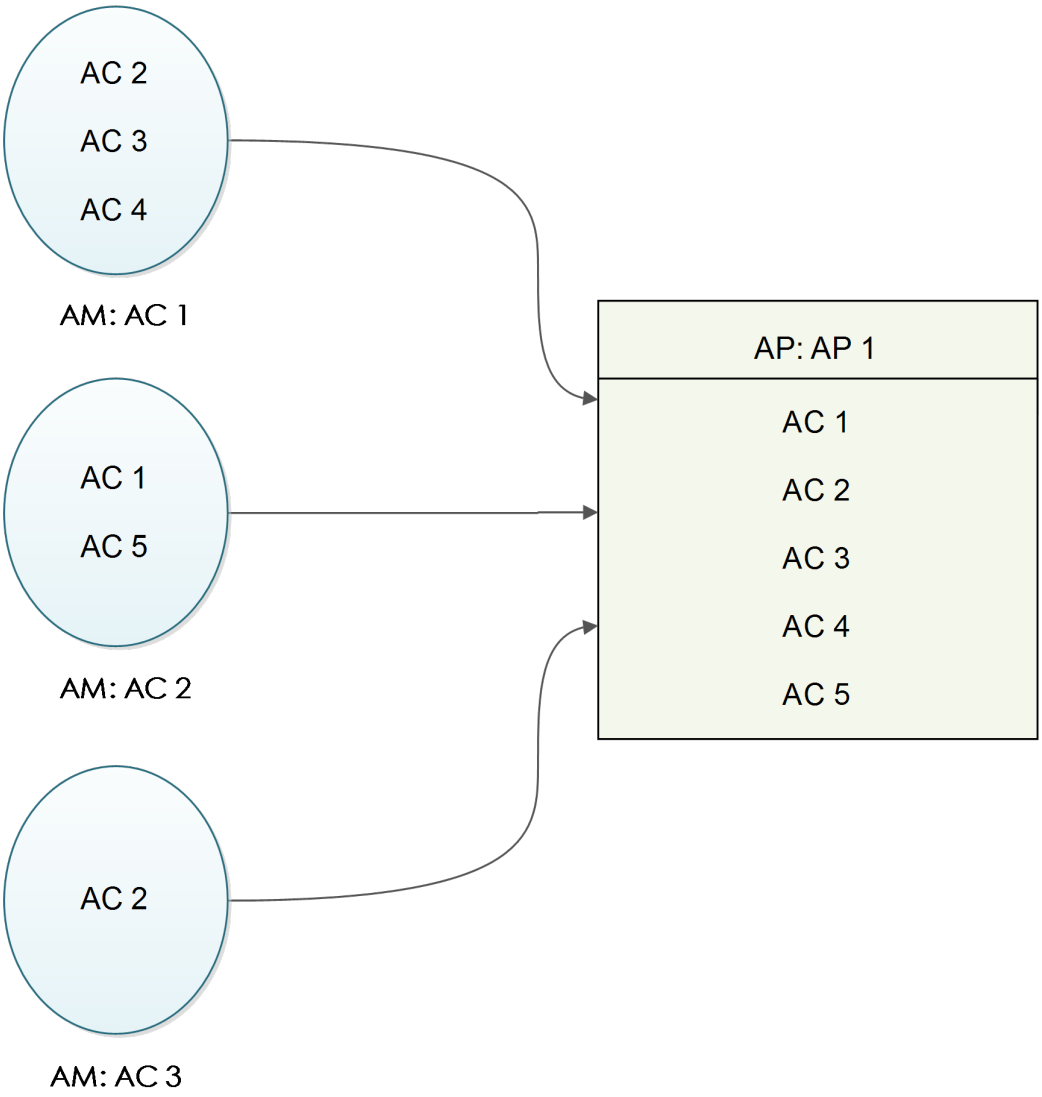


Figure 3.6: Profile Aggregation Example

Chapter 4

Experiments and Results

4.1 Environment and Programming Language

Our datasets are stored inside MarkLogic NoSQL database[25]. Basically, we used XQuery programming language[38] to develop our algorithm. In addition, we used different tools and libraries provided by MarkLogic server such as its power-full querying and searching functions as well as its different flexible and efficient indexing techniques. Moreover, we used MarkLogic task server[26] in order to parallelize the tasks in the three different phases of the algorithm.

To run our experiments, we used both a single server with a single node as well as a cluster environment with three power-full physical nodes. Table 4.1 shows the hardware specifications of the used environment.

Table 4.1: Environment Specifications

Server	Operating System	CPU Info	Memory	Disk Space
Cluster (3 nodes)	RHEL 6.5 (x86_64)	3 X 32 cores	3 X 96.7 GB	3 X 4.5 TB
Server (single node)	RHEL 6.6 (x86_64)	24 cores	84 GB	7 TB

4.2 Experiments

Our algorithm uses different parameters (thresholds and weights) that need to be tuned. Therefore, we conducted different experiments to tune these parameters as will be shown in the following subsections.

4.2.1 String Score Experiments

In this experiment, we aim at finding the best cutoff string score threshold. To do so, we ran the algorithm using only string score as matching criteria and we disabled all other stages of the algorithm. We repeated this experiment five times using five different string score thresholds (0.75 to 0.95). We used the *Selective* dataset to find the best threshold for string score.

According to the results of this experiment shown in Figure 4.1, the best cutoff threshold is (90%) where the highest complete profiles rate and the lowest error rate can be achieved. It is clear that higher string score threshold, we get high true positive rate but with a high false positive rate too. On the other hand, low string score produces a lower error profiles rat but with lower complete profiles rate.

Even that our string scoring algorithm is powerful and is able to detect most of the candidates, we found some cases where our chosen cutoff threshold(90%) failed to catch up some real candidate cases for a given name string. This is mainly due to different name string lengths in addition to names with initials. In order to account for these cases, we added another check for name consistency. The name consistency checker gives the opportunity for each name variation to be a candidate for the source author name even if the string score threshold is below 90%.

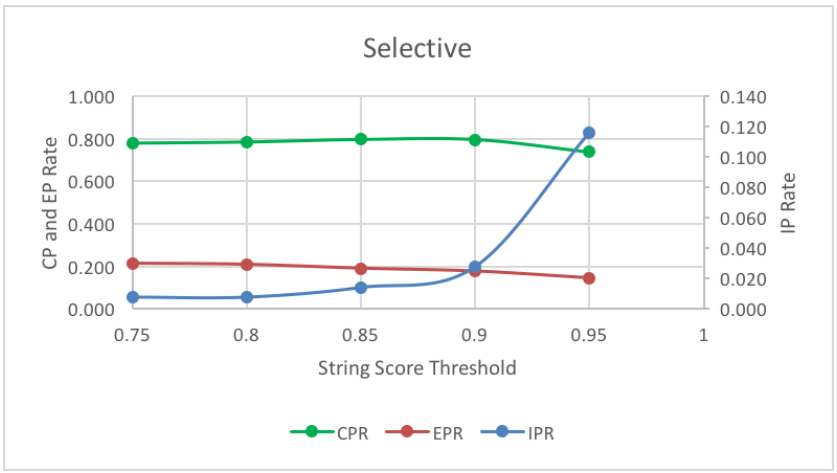


Figure 4.1: String Score Experiment

4.2.2 Matching Threshold Experiments

The result of comparing two ACs is a similarity score that represents how much the two records are close to each other. In order to decide that the two ACs belong the same individual, their similarity score should be above a certain threshold. This threshold is called matching threshold.

In order to find the best matching threshold, we applied our algorithm on the reference datasets using different thresholds. Figure 4.2 below represents the results using these different threshold. According to the result of this experiment, it is clear that 40% threshold is the best threshold that maximizes the CPR while minimize both IPR and EPR.

In addition, this experiment shows that high thresholds which are higher than 50% tend to produce a larger number of incomplete profiles at the expense of complete profiles while decreasing the number of error profiles. On the other hand, low thresholds less than 30% leads to higher number of error profiles and smaller number complete profiles.

4.2.3 Algorithm Speedup

Suppose that our database contains N different LNFI clusters and each LNFI cluster size is k_i, then the number of different all-against-all comparisons can be calculated using the equation 4.2.1. Since we used the LNFI to limit the search scope, the complexity of the algorithm depends on both the number of different clusters and the size of each cluster.

$$\text{Number of Comparisons} = \sum_{i=1}^{N} k_i(k_i - 1) \tag{4.2.1}$$

In our case, we have all-against-all comparisons within each cluster. Thus, for a given LNFI cluster of size k, we need $k(k-1)$ of AC comparisons. To speed up the algorithm, we used the following techniques:

- Parallel Execution
  The main three components have been implemented to run in parallel. Thus, the time needed is reduced by the degree of parallelism and the available concurrent running processes(threads).

- Code Optimization
  The main purpose of code optimization is to eliminate unneeded comparisons which either repeated or can be inferred using transitivity

property. We assumed that comparing two contributions is commutative/symmetric. For instance, if AC_1 is compared to AC_2, then when AC_2 turn comes, it never compared again with AC_1. Using the commutativity property, we reduced the number of comparisons to the half (from $k_i(k_i - 1)$ to $(k_i(k_i - 1))/2$).

In order to measure the impact of our improvements on the algorithm running time, we conducted some experiments against the *Selective* dataset which consists of *4466* contribution records that need to be disambiguated. We ran our algorithm against this dataset one time without our code optimization and a second time with our optimization. In order to eliminate the cache effect or any unexpected factors, we repeated each experiment two times and we calculated the average running time.

In addition, to test the impact of parallelism, we conducted two of these experiments using single-thread and two experiments using multi-threads(20 threads). The result of these experiments is shown in Figure 4.3. It is clear from Figure 4.3 that when we enable our code optimization techniques described previously, we can reduce the running time from about *41 minutes* to about *23 minutes* with about *42%* of time reduction. When we enable both the code optimization and the parallelism we reduced the running time noticeably to only about *3 minutes*.

These experiments were conducted on the single-node server described in Table 4.1. It is important to note that we only measured the running time for the second phase of the algorithm which is the slowest phase. The running time of the two other phases is negligible when compared to the time needed for the second phase.

Moreover, Figure 4.4 depicts the difference between the resulted sample AP with and without code optimization. Each node in the figure represents a contribution record and the arrow represents a comparison between the two records. The result shown in the figure shows that we can reach the same output profile but with much smaller number of comparisons when the code optimization is enabled.

4.3 Algorithm Results

In this section, we show the results of our ground-truth reference datasets. Also, Figure 4.13 shows the average results of all datasets.

4.3.1 Results of Abbas Dataset

This dataset represents all authors with last name Abbas and all their co-authors. It contains 1652 author name strings (including homonyms). The dataset was manually labeled to cluster the distinct individual authors. The number of profiles produced from the manual process is 1183 individual author profiles.

The algorithm produced 1226 profiles. Figure 4.5 shows that the absolute disambiguation accuracy for this dataset is 93.39%. That means the content of 93.39% of the generated profiles match exactly the result of the human generated profiles. It also shows that 6.2% of the generated profiles are correct but divided into more than one profile. For example, an individual might have his/her data divided in more than one profile, but this data is not mixed with other individuals' data. 0.41% of the generated profiles contain the exact data for an individual but it has extra data from other individuals. Finally, we have 0.0% IER which means that we do not have mixed incomplete data for different individuals.

4.3.2 Results of Mahmood Dataset

This dataset represents all authors with last name Mahmood and all their co-authors. It contains 1259 author name strings (including homonyms). The dataset was manually labeled to cluster the distinct individual authors. The number of profiles produced from this process is 886 individual author profiles.

The algorithm produced 914 profiles. Figure 4.6 shows that the absolute disambiguation accuracy for this dataset is 93.54%. That means the content of 93.54% of the generated profiles match exactly the result of the human generated profiles. It also shows that 5.47% of the generated profiles are correct but divided into more than one profile. For example, an individual might have his/her data divided in more than one profile, but this data is not mixed with other individuals' data. 0.77% of the generated profiles contain the exact data for an individual but it has extra data from other individuals. 0.22% has mix data for different individuals.

4.3.3 Results of Collet Dataset

This dataset represents all authors with last name Collet and all their co-authors. It contains 2885 author name strings (including homonyms).

The dataset was manually labeled to cluster the distinct individual authors. The number of profiles produced from this process is 1324 individual author profiles.

The algorithm produced 1417 profiles. Figure 4.7 shows that the absolute disambiguation accuracy for this dataset is 89.06%. That means the content of 89.06% of the generated profiles match exactly the result of the human generated profiles. It also shows that 10.66% of the generated profiles are correct but divided into more than one profile. For example, an individual might have his/her data divided in more than one profile, but this data is not mixed with other individuals' data. 0.28% of the generated profiles contain the exact data for an individual but it has extra data from other individuals. 0.0% has mix data for different individuals.

4.3.4 Results of Vail Dataset

This dataset represents all authors with last name Vail and all their co-authors. It contains 2545 author name strings (including homonyms). The dataset was manually labeled to cluster the distinct individual authors. The number of profiles produced from this process is 1304 individual author profiles.

The algorithm produced 1366 profiles. Figure 4.8 shows that the absolute disambiguation accuracy for this dataset is 92.53%. That means the content of 92.53% of the generated profiles match exactly the result of the human generated profiles. It also shows that 7.17% of the generated profiles are correct but divided into more than one profile. For example, an individual might have his/her data divided in more than one profile, but this data is not mixed with other individuals' data. 0.29% of the generated profiles contain the exact data for an individual but it has extra data from other individuals. 0.0% has mix data for different individuals.

4.3.5 Results of Racine Dataset

This dataset represents all authors with last name Racine and all their co-authors. It contains 1426 author name strings (including homonyms). The dataset was manually labeled to cluster the distinct individual authors. The number of profiles produced from this process is 673 individual author profiles.

The algorithm produced 707 profiles. Figure 4.9 shows that the absolute disambiguation accuracy for this dataset is 90.24%. That means the content of 90.24% of the generated profiles match exactly the result of the human generated profiles. It also shows that 7.92% of the generated profiles are correct but divided into more than one profile. For example, an individual might have his/her data divided in more than one profile, but this data is not mixed with other individuals' data. 1.41% of the generated profiles contain the exact data for an individual but it has extra data from other individuals. 0.42% has mix data for different individuals.

4.3.6 Results of Royer Dataset

This dataset represents all authors with last name Royer and all their co-authors. It contains 1239 author name strings (including homonyms). The dataset was manually labeled to cluster the distinct individual authors. The number of profiles produced from this process is 736 individual author profiles.

The algorithm produced 786 profiles. Figure 4.10 shows that the absolute disambiguation accuracy for this dataset is 90.2%. That means the content of 90.2% of the generated profiles match exactly the result of the human generated profiles. It also shows that 9.8% of the generated profiles are correct but divided into more than one profile. For example, an individual might have his/her data divided in more than one profile, but this data is not mixed with other individuals' data. 0% of the generated profiles contain the exact data for an individual but it has extra data from other individuals. 0% has mix data for different individuals.

4.3.7 Results of Selective Dataset

This dataset represents an extreme case of author name strings. It contains 4466 author name strings (including homonyms). We tried to collect names from multiple backgrounds and common last names. The data is collected based on the following filters Brown W, Xue H and Ramos M. The dataset was manually labeled to cluster the distinct individual authors. The number of profiles produced from this process is 2498 individual author profiles.

The algorithm produced 2666 profiles. 4.11 shows that the absolute disambiguation accuracy for this dataset is 89.2%. That means the content of

89.2% of the generated profiles match exactly the result of the human generated profiles. It also shows that 9.71% of the generated profiles are correct but divided on more that one profile. For example, an individual might have his/her data divided into more than one profile, but this data is not mixed with other individuals' data. 0.75% of the generated profiles contain the exact data for an individual but it has extra data for other individuals. 0.34% has mix data for different individuals.

4.3.8 Results of Zang Dataset

This dataset represents all authors with last name Zang and all their co-authors. It contains 3539 author name strings (including homonyms). The dataset was manually labeled to cluster the distinct individual authors. The number of profiles produced from this process is 2040 individual author profiles.

The algorithm produced 2170 profiles. 4.12 shows that the absolute disambiguation accuracy for this dataset is 90.09%. That means the content of 90.09% of the generated profiles match exactly the result of the human generated profiles. It also shows that 8.48% of the generated profiles are correct but divided into more than one profile. For example, an individual might have his/her data divided in more than one profile, but this data is not mixed with other individuals' data. 1.24% of the generated profiles contain the exact data for an individual but it has extra data from other individuals. 0.18% has mix data for different individuals.

4.3.9 Average Result

The average result across all the datasets above is shown in Figure 4.13. The average absolute disambiguation accuracy is 91.03%. That means the content of 91.03% of the generated profiles match exactly the result of the human generated profiles. It also shows that 8.18% of the generated profiles are correct but divided into more than one profile. For example, an individual might have his/her data divided in more than one profile, but this data is not mixed with other individuals' data. 0.64% of the generated profiles contains the exact data for an individual but it has extra data from other individuals. 0.15% has mix data for different individuals.

4.3.10 Additional Datasets Results

To ensure the generalization of the algorithm, we used three more datasets (*Peters E*, *Paul S* and *Zhang M)* that have been labeled manually. These unseen datasets which were hidden during our algorithm design and thresholds selection can give an insight about the generalization of the algorithm. The description of these datasets is shown in Table 3.2. The results of these datasets are shown in Figures 4.14, 4.15 and 4.16.

The results of these datasets are consistent with the results of our reference datasets which prove the ability of our algorithm to generalize on new datasets with high accuracy. It is clear that we can get more than 90% of complete profiles that exactly match the manually disambiguated profiles. Also, the incomplete profiles ratio for the three datasets (8.96%, 3.87% and 4.69%) is within the expected range.

In addition, we can see that both *Paul S* and *Peters E* achieved very low error profiles ratios (0.59% and 0.86% respectively). However, *Zhang M* dataset showed a higher portion of error profiles which refers to two main reasons. First, this dataset represents Chines names which are very challenging. Second, this dataset is more prone to human error since it is larger in size and contains a higher level of ambiguity.

4.4 Algorithm Generalization

We applied our algorithm against the entire PsycINFO database in order to test the generalization and the scalability of the algorithm in a real environment. PsycINFO database contains about 10.5 million author name instances including homonyms. The algorithm produced about 4 million author profiles with a ratio of 2.58 contributions per profile as shown in Table 4.2.

In order to evaluate the results of the algorithm, we conducted two types of experiments. The main objective of these experiments is to assess the accuracy of the results and to evaluate the generalization of the algorithm. In the first experiment we try to detect candidate error profiles while in the second experiment we estimate the portion of incomplete profiles.

The disambiguation process of entire database is performed using a cluster environment that consists of three powerful multi-core physical nodes.

The specification of the running environment is described in Table 4.1. The algorithm took about 80 hours to finish on the entire database. Most of the time (80% of the total time) has been taken by the second phase where the disambiguation decision is taken.

Table 4.2: The algorithm results of the entire PsycINFO database

AC Records	AP Records	AC/AP Rate
10,670,921	4,133,094	2.58

4.4.1 Candidate Error Profiles

In this experiment, we assumed that any AP record that contains any two name string variations with string score below a given threshold as a candidate error profile where two different authors have been merged in the same profile. We repeated this experiment twice, each time, we used a different string threshold (85% and 90%). The result of this experiment is shown the table 4.3. It is clear that the error rate using both thresholds is less than the average error rate reported using the ground truth datasets.

Table 4.3: Candidate Error Profiles

Total AP Count	String Score Threshold	Error Profile Candidates	Error Profiles Percentage
4,133,094	85%	6,893	1.67E-03
	90%	1,4768	3.57E-03

4.4.2 Candidate Partial Profiles

Partial profiles are correct but incomplete profiles. This case occurs when two or more contributions of the same person are split into two or more profiles. This experiment is based on the assumption that rare(less frequent) name strings are more likely to refer to the same person. If the exact name strings of rare names that come from different ACs have been distributed among two or more profiles, then these profiles are counted as incomplete profiles.

In this experiment, we used only full names and we excluded names with initials. We repeated the experiment different times using different definition for low frequency names. The results are shown in Table 4.4. Candidate partial profiles with name frequency greater than 10 is 322,170 (0.078 of total

profiles). The total candidate partial profiles regardless of name frequency is 726,855(0.176 of total profiles).

Table 4.4: Candidate Partial Profiles

AP Records	Name Frequency(<=)	Candidate Partial Profiles	Partial Profiles Ratio
4,133,094	2	122,812	0.030
	4	258,642	0.063
	6	329,808	0.080
	8	373,704	0.090
	10	404,685	0.098

4.4.3 ORCID Experiment

ORCID (Open Researcher and Contributor ID) is a unique digital identifier used to distinguish researchers, contributors and academic authors[31]. Once the author creates his ORCID, he can use it as global ID for his contributions/publications.

In this experiment, we used the ORCID to evaluate the reslults our algorithm. There are about 60,000 ORCID ID inside our database. We compared the results of the algorithm (algorithm profiles) with existing ORCID profiles. ORCID profile is an ORCID record that contains all publications of a given author.

We have two cases for the ORCID profile. In the best case, ORCID profile is mapped to a single corresponding algorithm profile. On the other hand, ORCID profile might be distributed among two or more algorithm profiles. The result of this experiment shows that about ∼92% of ORCID profiles, either have only a single profile(∼79%) or have been split into only two profiles(∼13%). Other cases represent only about ∼8% of the ORCID Profiles. Figure 4.17 depicts the partials rates.

4.5 Discussion

Our results indicate that the highest portion of output profiles (91% on average) are complete profiles that exactly match their corresponding profiles in the reference datasets. The rest of profiles are distributed between partially correct and error profiles.

We have, on average, about 8% of incomplete profiles that partially match their corresponding profiles. The incomplete profiles are pure profiles, i.e. they contain only contributions of the same individual but some of the contributions are missed. In these cases the author/contributor will find his contributions divided into two or more profiles. The last portion of output profiles represents the error profiles. We have less than 0.8% of error profiles which represent a very small portion of the output profiles.

The worst type of errors is IE where a subset of contributions of two or more individuals is falsely merged together. According to the results, we were able to minimize the IER so that it is the minimum rate among other types. For the datasets *Racine*, *Vail*, *Collet*, *Abbas* and *Peters E*, IER is 0.0%. For other datasets, the maximum observed IER is 0.53% in *Zhang M* dataset.

The minimum achieved CPR is around 89% for both *Selective* and *Collet* datasets. The *Selective* dataset represents an extreme dataset in terms of size and names diversity since it was selected using three different names. On the other hand, *Collet* dataset contains many of abbreviated full names using name initials which increased the level of ambiguity in this dataset.

There is a clear trade-off between CP profiles and other types of profiles, especially the IP profiles. In order, to minimize the error profiles, the algorithm tends in certain cases to be more conservative which causes splitting some of CP profiles into two or more IP profiles and thus increasing the IPR. This behavior can be observed in different dataset that achieved around 90% such as *Royer* dataset. In those datasets, we observe a higher IPR (7%- 8%).

The high IPR is due to several reasons such as the absence of some features and very ambiguous cases where the disambiguation decision is hard to take. In these cases, the algorithm tends towards segregating the two compared contributions rather than aggregating them together. Our assumption is that, it is more acceptable for the author to find his contributions splitted into two profiles than to find some of his contributions falsely merged within profiles of other authors.

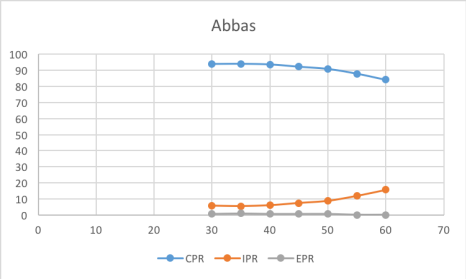
Comparing our results to previous AD methods is difficult due to the absence of a common benchmark dataset. In fact, constructing such a benchmark dataset is a challenging task, especially that the AD is context dependent problem. However, our algorithm is distinguished by a set features including efficiency, adaptivity and flexibility.

In contrast to other approaches, using our heuristic approach, we are able to model real life models such as professor-students model and author movements model. This flexibility enhances the accuracy of the results. In addition, the heuristic approach requires no training data which represent the main challenge for supervised AD methods.

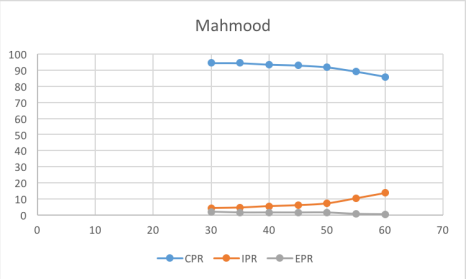
During our literature review, we observed the absence of a quantitative assessment of the proposed approaches in terms efficiency. In this work, we conducted some experiments to quantitatively measure the computation time using *Selective* dataset. The results showed that we are able to reduce the disambiguation time noticeably with about 42% of time reduction. This reduction in time was obtained due to many speedup techniques in addition to the parallel execution of tasks within each phase of algorithm.

One of the main speedup techniques is the multi-stage matching logic implemented in the second phase of the algorithm. This technique is based on the assumption that, in some cases we can take the disambiguation decision using a only subset of features. We move to the next stage only if we are not able to take the decision using the features score in the current stage. In addition, we implemented some optimization techniques to reduce the number of performed comparisons by leveraging some properties such as transitivity and commutativity.

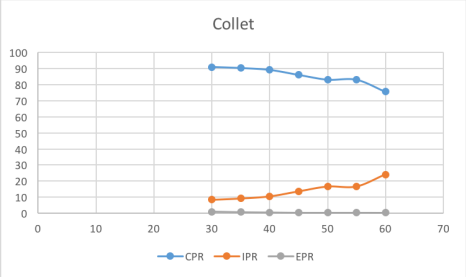
Figure 4.2: Silver Stage Threshold Selection



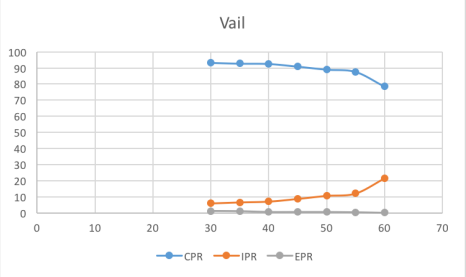
(a) Abbas



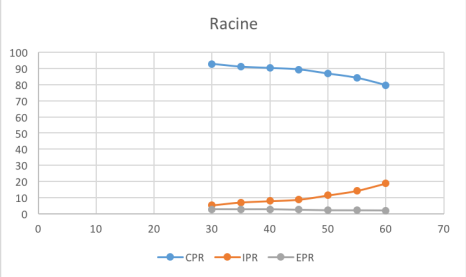
(b) Mahmood



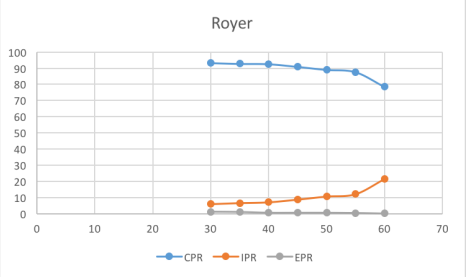
(c) Collet



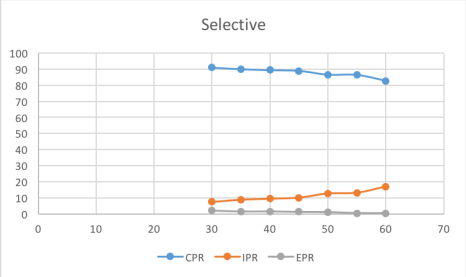
(d) Vail



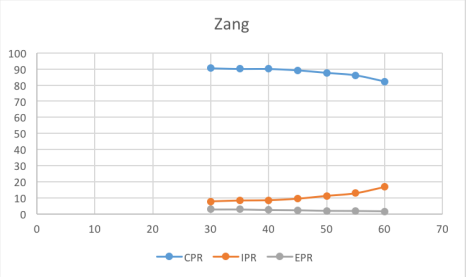
(e) Racine



(f) Royer



(g) Selective



(h) Zang

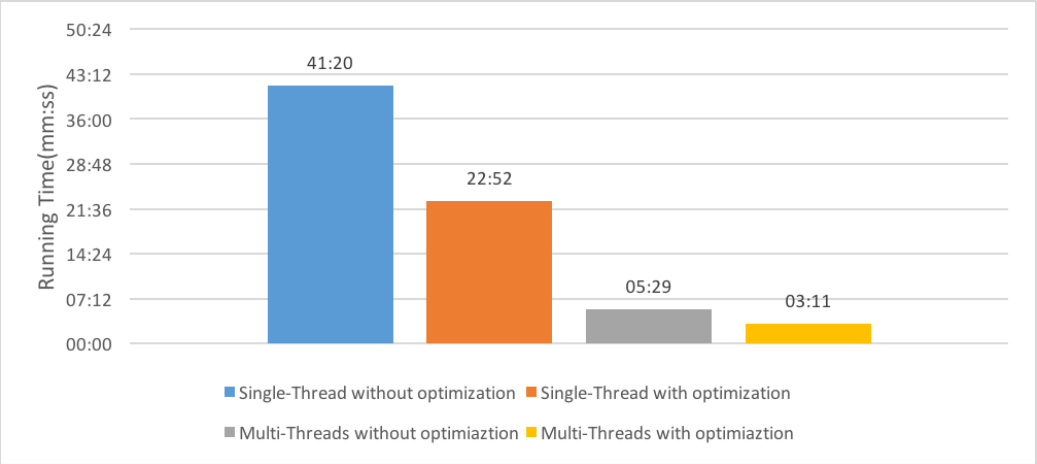
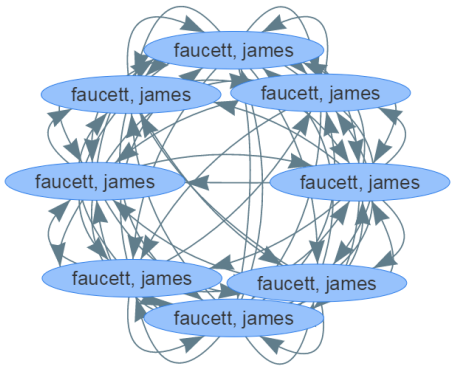
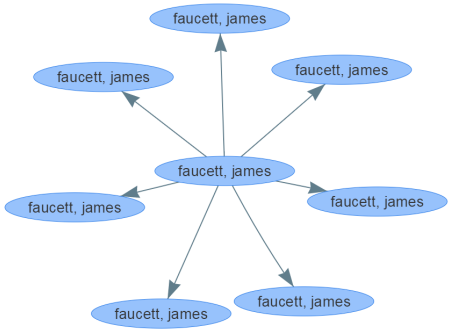


Figure 4.3: Algorithm Speedup

Figure 4.4: Sample AP with/without optimization



(a) Sample AP without optimization



(b) Sample AP with optimization

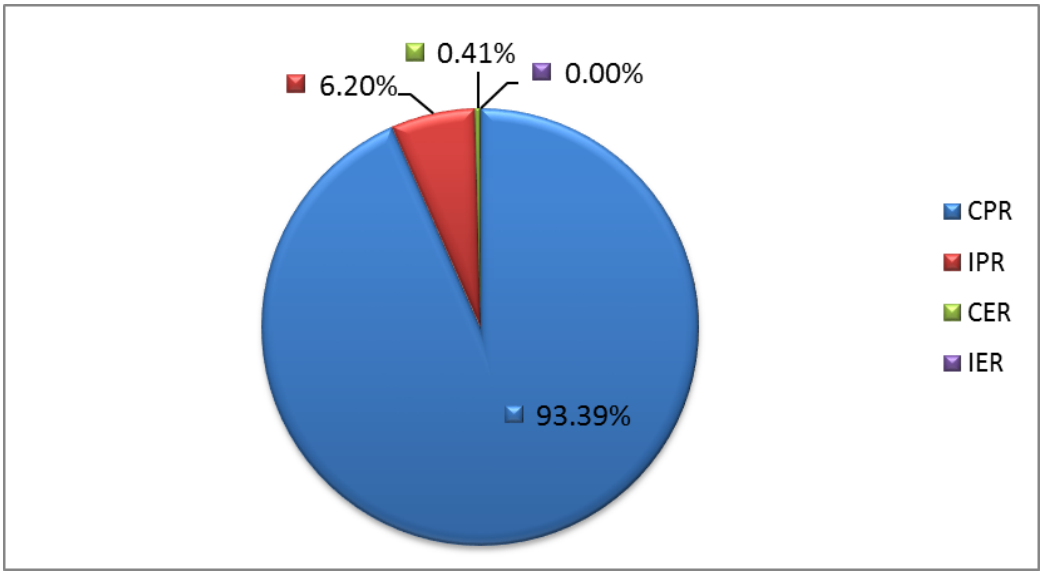


Figure 4.5: Algorithm result for Abbas dataset

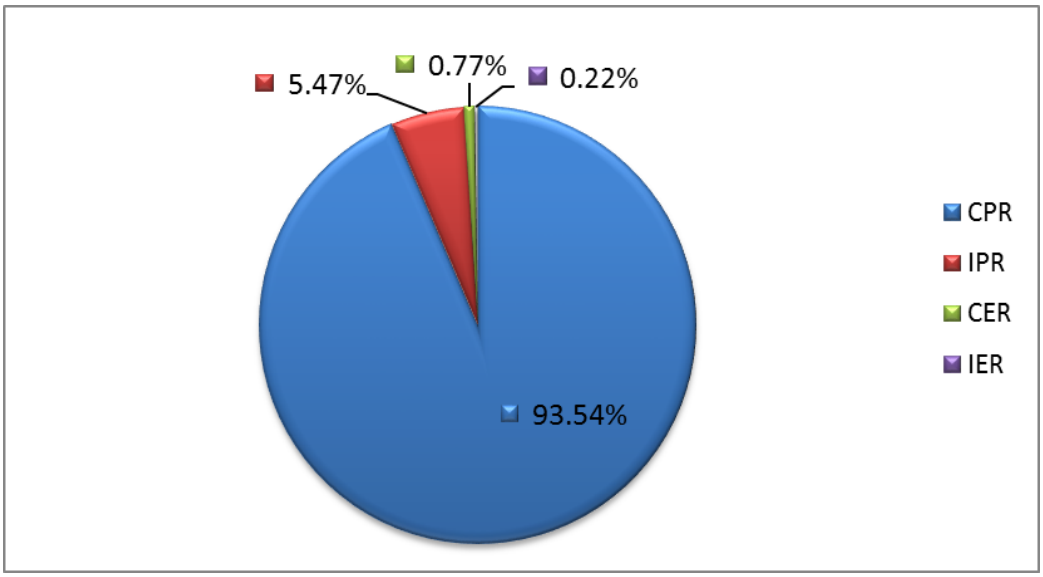


Figure 4.6: Algorithm result for Mahmood dataset

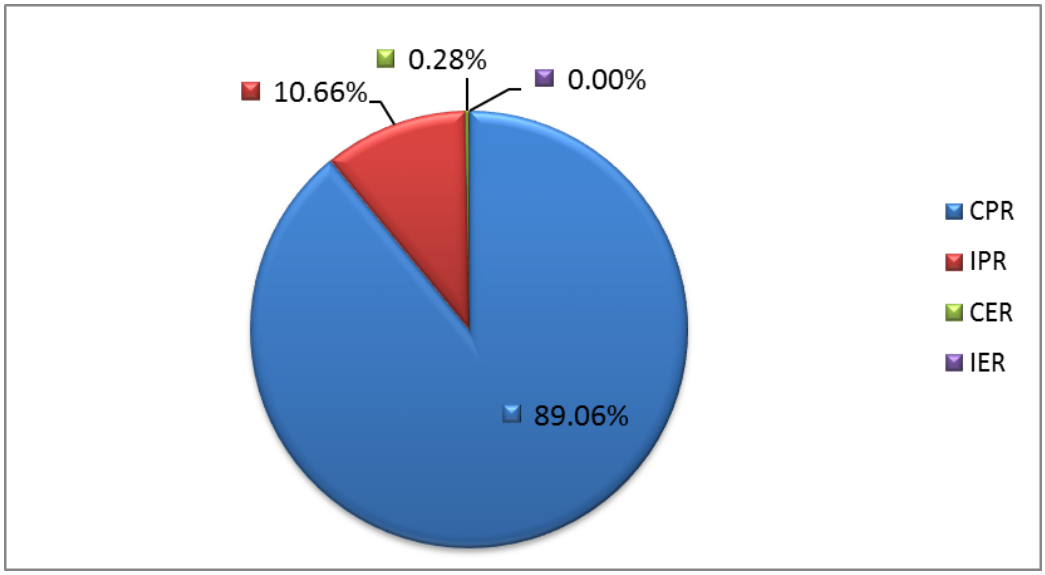


Figure 4.7: Algorithm result for Collet dataset

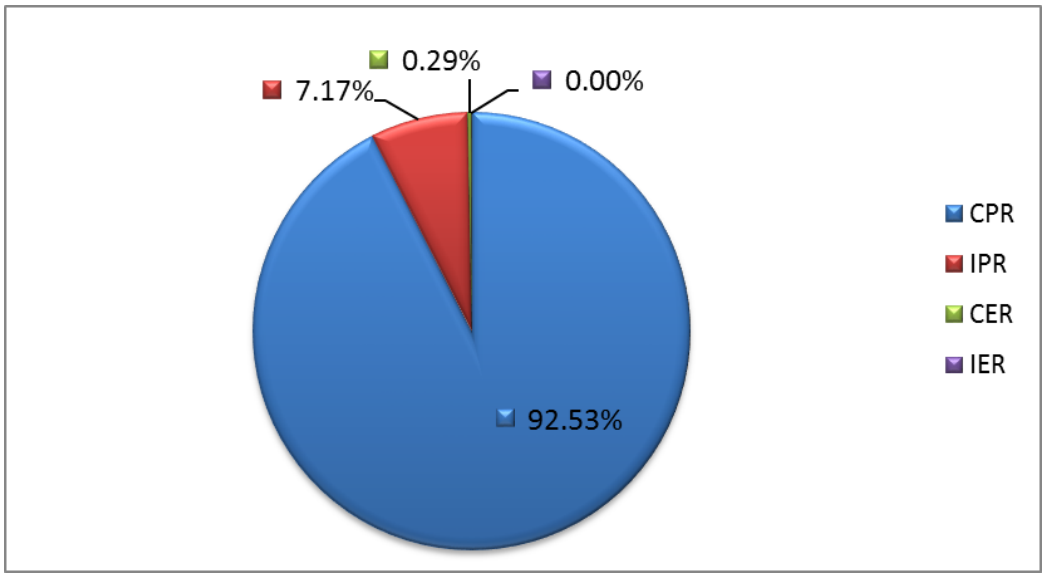


Figure 4.8: Algorithm result for Vail dataset

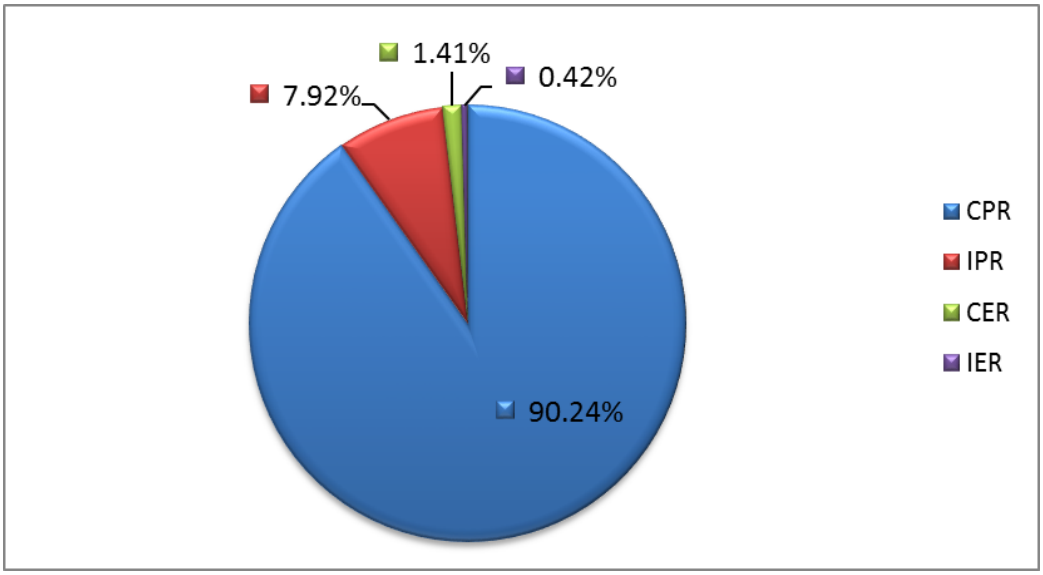


Figure 4.9: Algorithm result for Racine dataset

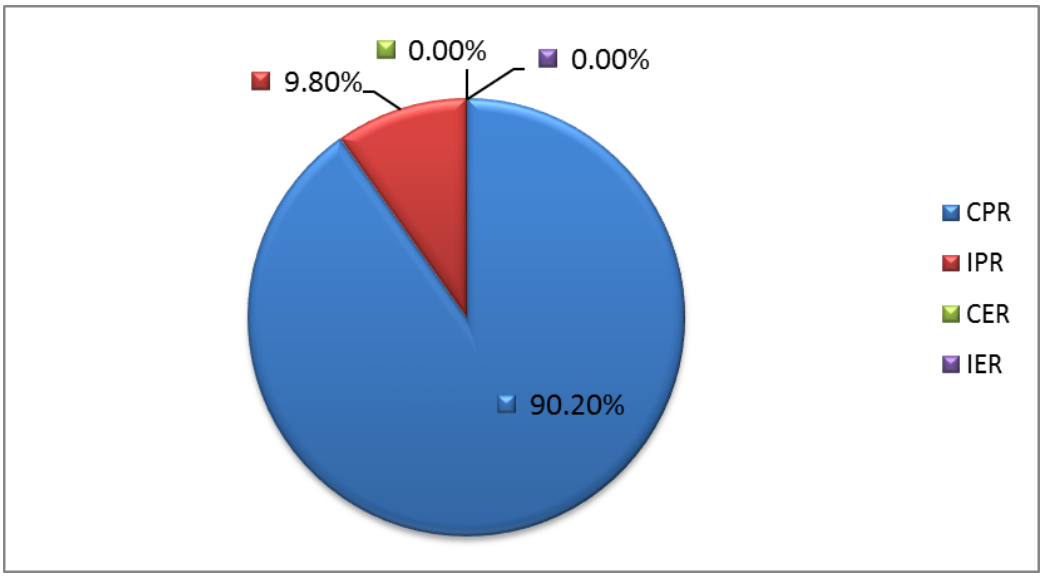


Figure 4.10: Algorithm result for Royer dataset

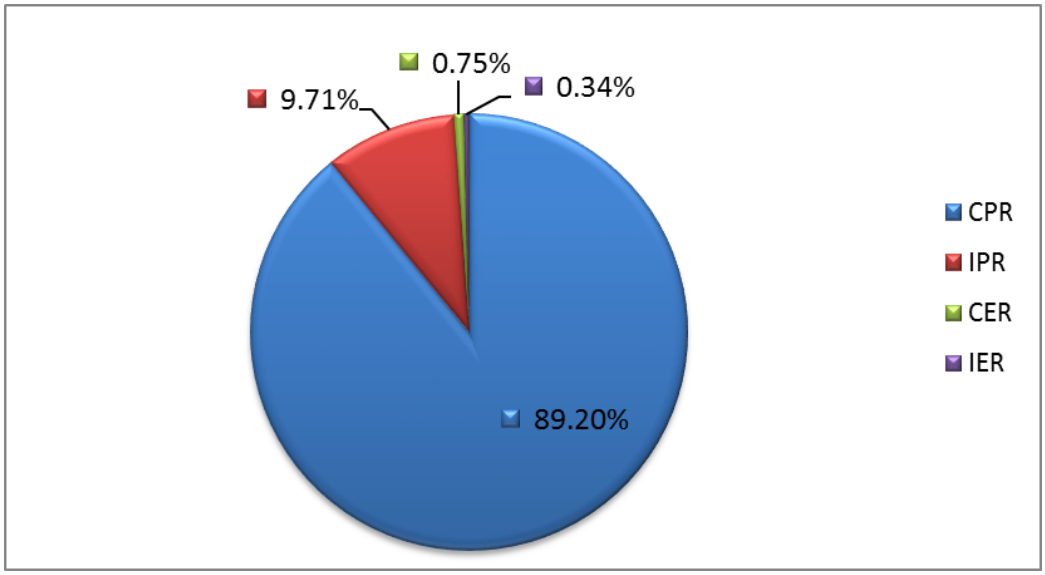


Figure 4.11: Algorithm result for Selective dataset

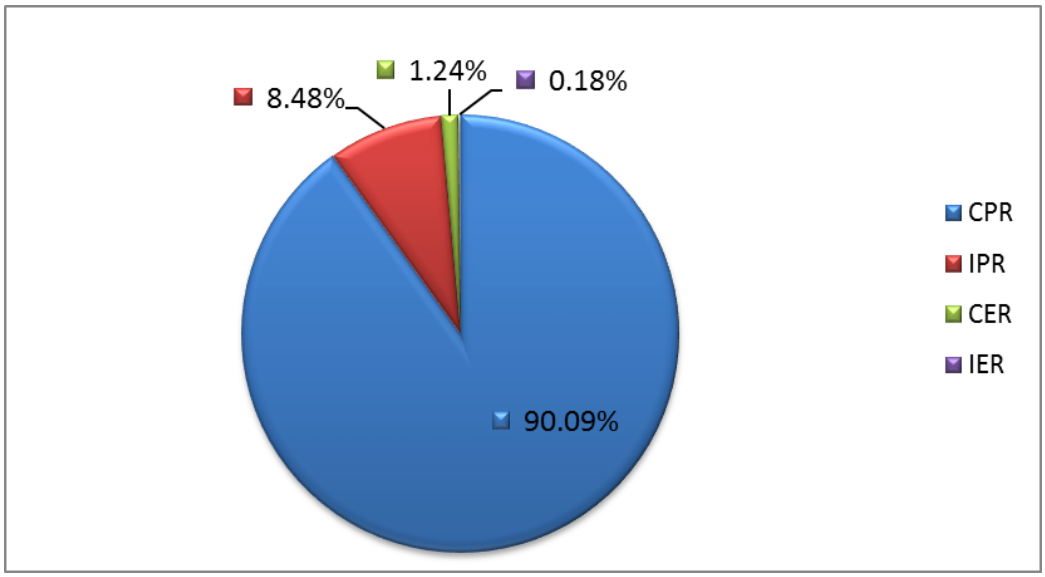


Figure 4.12: Algorithm result for Zang dataset

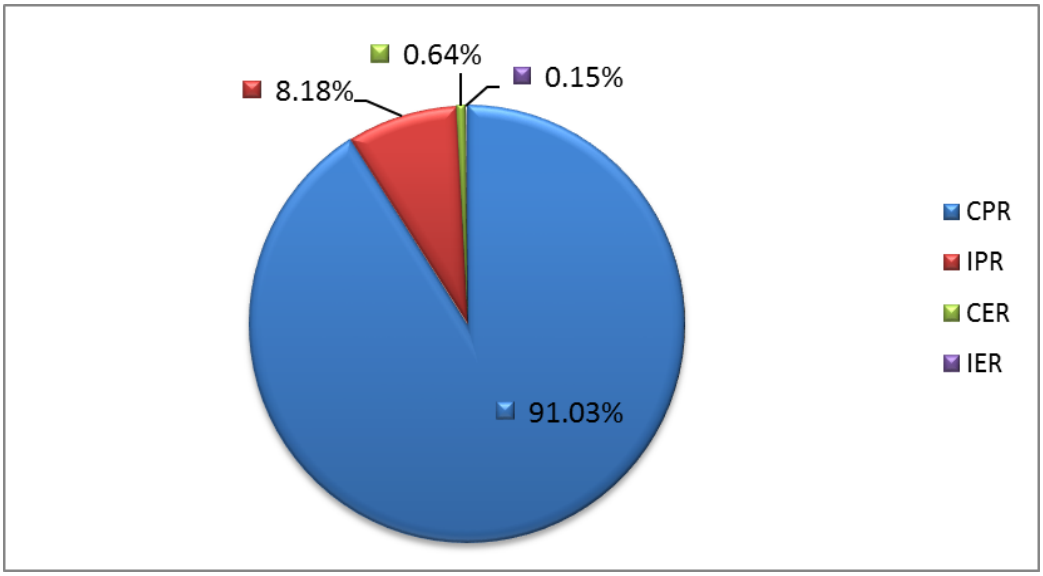


Figure 4.13: Algorithm Average Result

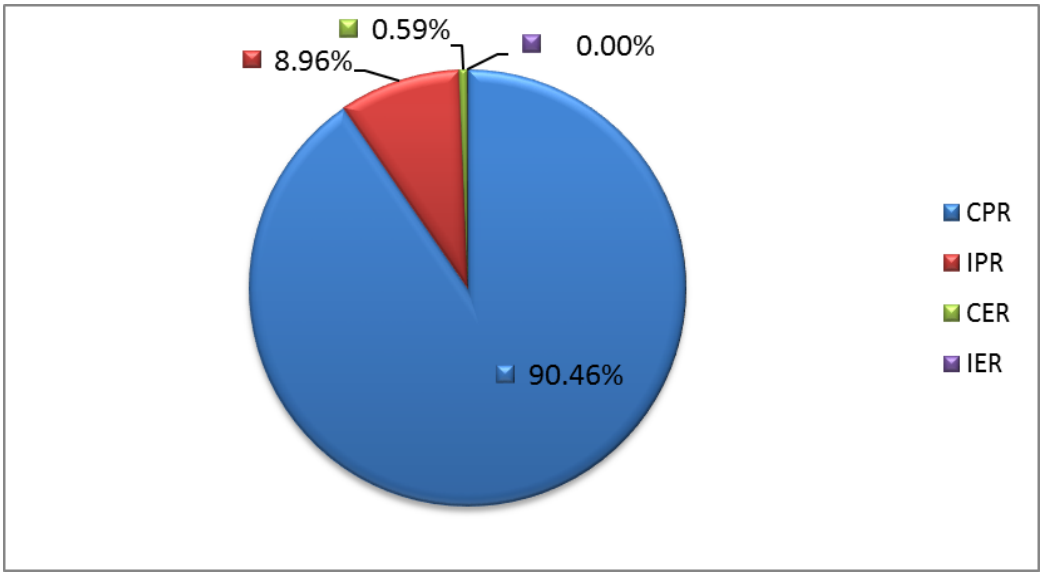


Figure 4.14: Peters, E Dataset

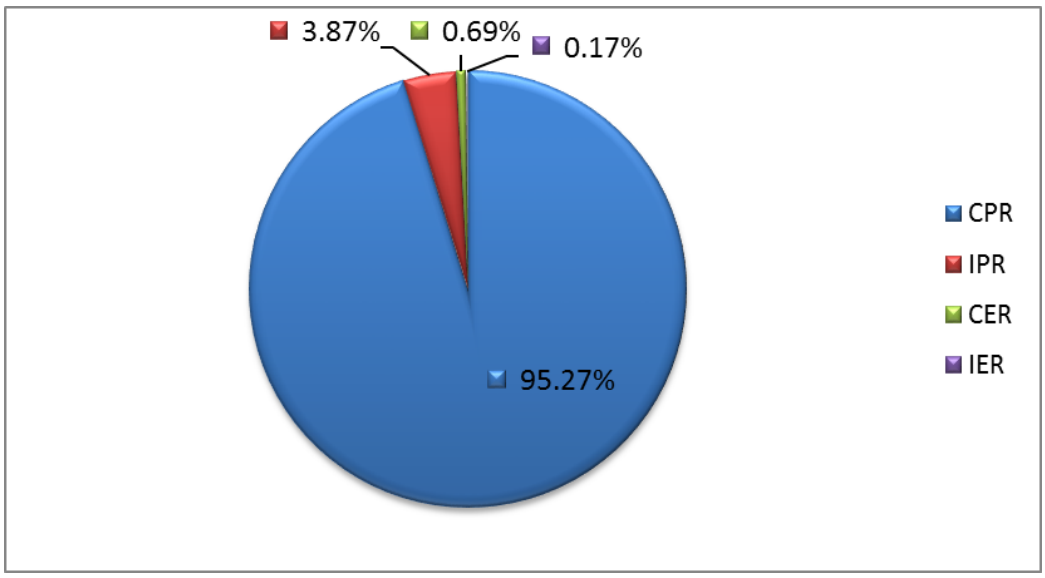


Figure 4.15: Paul, S Dataset

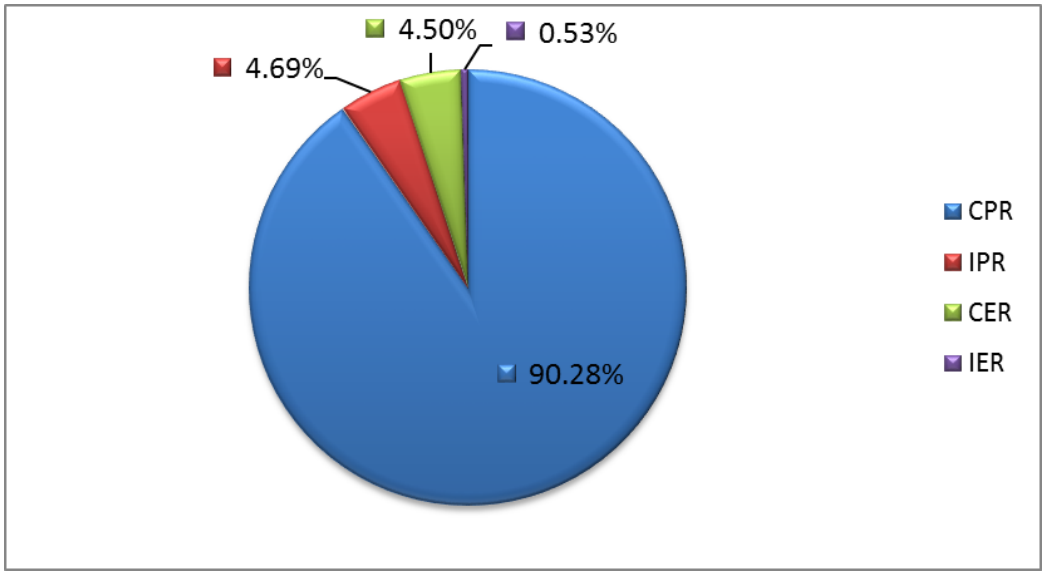


Figure 4.16: Zhang, M Dataset

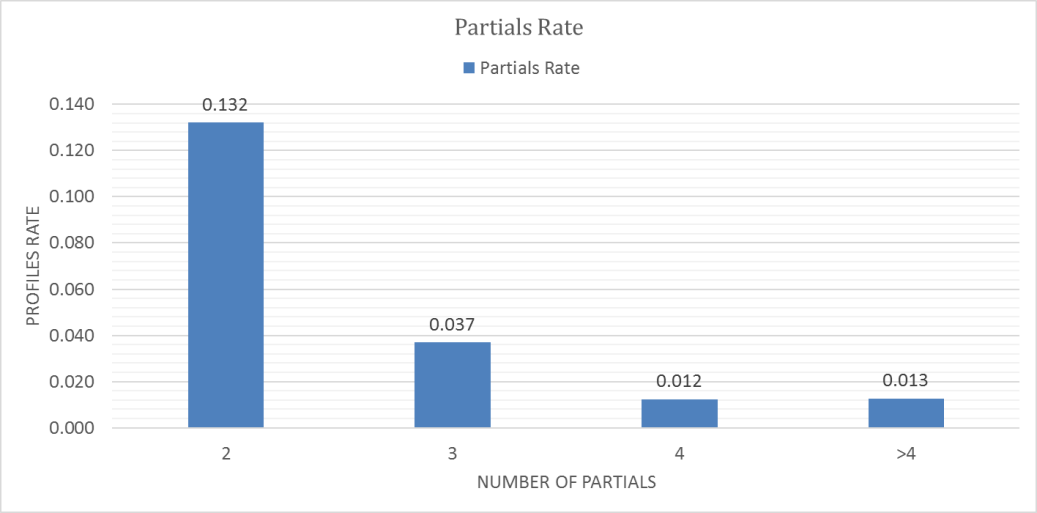


Figure 4.17: ORCID Partials Rates

Chapter 5

Conclusion and Future Work

5.1 Conclusion

Solving author name ambiguities can be helpful in many different applications especially for digital libraries. Creating author profiles, identifying the contributions of authors and studying their collaboration networks are clear examples where AD is beneficial.

In this work, we proposed a new heuristic-based algorithm to address AD problem. Our proposed algorithm is designed to be efficient, scalable, adaptive and accurate.

Unlike supervised methods, our heuristic approach does not require any training data. Preparing training data is challenging task since the training examples need to be both representative and sufficient in terms of quantity and quality.

Our algorithm is decomposed into three different phases. Each phase is designed so that all its tasks can be run in parallel. In addition, the second phase which implements the matching logic is designed to work in stages in order to take the decision as fast as possible.

Using parallelism and multi-stage decision making in addition to some optimization techniques, the proposed algorithm is proved to be efficient and scalable to work in a real environment where there are millions of name instances need to be disambiguated within a relatively short time.

The proposed algorithm is enriched with different adaptivity features such

as considering name popularity/rarity in the disambiguation process. This adaptive behavior enabled the algorithm to cover the wide variations in a large collection of name strings.

Using our heuristic approach allows for tuning the algorithm to account for the causes of errors, and also allows for the many benefits of the intelligent clustering up front to ensure both the accuracy and the efficiency of the algorithm. However, the heuristics do rely on reactions to observed conditions and errors, and is thus reliant on the characteristics of the immediate dataset and the conclusions of the observer.

We validated our algorithm against a collection of manually labeled reference datasets that cover a wide range of variations. These datasets were collected from APA PsycINFO database. The results showed that about 91% of the output profiles exactly matched the profiles in the reference datasets and about 8.18% partially matched the reference profiles and only less than 0.8% of error profiles.

To prove the scalability, we applied our algorithm against the entire APA PsycINFO database that contains more than 4 million publications with about 10.5 million name string instances including homonyms. The disambiguation of all authors in the whole database was done with in 4 days which is a relatively short time. Our assessment of this process indicates that the achieved accuracy and error rates are consistent with the reference datasets results.

According to our results, we conclude that the heuristic based approach can be used to handle AD problem efficiently with a high accuracy results. In addition, our results showed that the heuristic approach is scalable and can be generalized to work in real large databases. Moreover, the heuristic approach is more controllable and flexible to model real life models and can be combined with some adaptive features to enhance the accuracy of the results.

5.2 Future Work

In spite of the high accuracy we obtained in our algorithm, it is clear that 100% accuracy can not be achieved using automatic disambiguation methods only. Therefore, we think that hybrid solution that combines both automatic disambiguation in addition to the human intervention is the best

effective approach to address the AD problem.

We think that combining our algorithm with some human intervention will lead to more accurate results and more user satisfaction. In this way, we can solve very ambiguous cases where the records of different authors are falsely merged together. On the other hand, the author can aggregate his records that the algorithm decided to split because of missing information or low similarity evidence.

It is important to note that our proposed algorithm is designed to work on a pre-existing database as a batch processing. However, current digital libraries continuously ingest new articles to their databases. Therefore, there is a need to disambiguate author names during the ingestion process. This type of disambiguation is called incremental disambiguation.

Incremental disambiguation should be efficient and fast since it will be performed on regular basis in a real environment. The incremental algorithm should be designed to deal with the new records without the need to re-disambiguate the entire database which will be expensive in terms of time and processing power. Certainly, the incremental algorithm would make use of the core logic and scoring methods developed in our algorithm.

During the disambiguation process, we used the transitivity property to aggregate similar contributions in the same record without performing the actual comparisons. Despite that transitive aggregation was a valuable optimization technique in our algorithm, we should denote that accepting the transitive result as an absolute fact might lead to some side-effects. Therefore, we think that it is valuable to investigate the effect of transitivity on the accuracy of the algorithm and try to handle its subtle problems.

Also, our algorithm depends on different thresholds to take different decisions. Basically, we conducted different experiments to choose the best thresholds. However, these experiments look at each threshold independently. Another way to do the selection, is to use one of the optimization tools that considers the different combination of parameters and selects the best thresholds.

In addition, our scoring methods are based only on a subset of features such as email, coauthors and affiliation. Our assumption was to focus on the most important features in order to minimize the disambiguation time and enhance the algorithm efficiency without noticeably affecting the accuracy of

the results. However, it is a good idea to explore some new features such as article title, keywords and citation overlap, and study their impact on both the performance of the algorithm as well as the accuracy of the results.

Bibliography

[1] APA. About apa. `http://www.apa.org/about/`, 2017. (Accessed on June, 2017).

[2] APA. Apa databases & electronic resources. `http://www.apa.org/pubs/databases/index.aspx`, 2017. (Accessed on June, 2017).

[3] APA. Psycinfo printable fact sheet. `http://www.apa.org/pubs/databases/psycinfo/psycinfo-printable-fact-sheet.pdf`, 2017. (Accessed on June, 2017).

[4] T Arif, R Ali, and M Asger. A multistage hierarchical method for author name disambiguation. *International Journal of Information Processing*, 9(3):92–105, 2015.

[5] Indrajit Bhattacharya and Lise Getoor. Collective entity resolution in relational data. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):5, 2007.

[6] Wei-Sheng Chin, Yong Zhuang, Yu-Chin Juan, Felix Wu, Hsiao-Yu Tung, Tong Yu, Jui-Pin Wang, Cheng-Xia Chang, Chun-Pai Yang, Wei-Cheng Chang, et al. Effective string processing and matching for author disambiguation. *The Journal of Machine Learning Research*, 15(1): 3037–3064, 2014.

[7] William Cohen, Pradeep Ravikumar, and Stephen Fienberg. A comparison of string metrics for matching names and records. In *Kdd workshop on data cleaning and object consolidation*, volume 3, pages 73–78, 2003.

[8] Ricardo G Cota, Marcos André Gonçalves, and Alberto HF Laender. A heuristic-based hierarchical clustering method for author name disambiguation in digital libraries. In *SBBD*, pages 20–34. Citeseer, 2007.

[9] Aron Culotta, Pallika Kanani, Robert Hall, Michael Wick, and Andrew McCallum. Author disambiguation using error-driven machine learning with a ranking loss function. In *Sixth International Workshop on*

Information Integration on the Web (IIWeb-07), Vancouver, Canada, 2007.

[10] Ana Paula de Carvalho, Anderson A Ferreira, Alberto HF Laender, and Marcos A Gonçalves. Incremental unsupervised name disambiguation in cleaned digital libraries. *Journal of Information and Data Management*, 2(3):289, 2011.

[11] Sarah Elliot. Survey of author name disambiguation: 2004 to 2010. 2010.

[12] Luciano Vilas Boas Esperidião, Anderson A Ferreira, Alberto HF Laender, Marcos André Gonçalves, David Menotti Gomes, Andrea Iabrudi Tavares, and Guilherme Tavares de Assis. Reducing fragmentation in incremental author name disambiguation. *Journal of Information and Data Management*, 5(3):293, 2014.

[13] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[14] Anderson A Ferreira, Marcos André Gonçalves, and Alberto HF Laender. A brief survey of automatic methods for author name disambiguation. *Acm Sigmod Record*, 41(2):15–26, 2012.

[15] Anderson A Ferreira, Marcos André Gonçalves, and Alberto HF Laender. Automatic methods for disambiguating author names in bibliographic data repositories. In *Proceedings of the 15th ACM/IEEE-CE on Joint Conference on Digital Libraries*, pages 297–298. ACM, 2015.

[16] Hui Han, Hongyuan Zha, and C Lee Giles. A model-based k-means algorithm for name disambiguation. In *Proceedings of the 2nd International Semantic Web Conference (ISWC-03) Workshop on Semantic Web Technologies for Searching and Retrieving Scientific Data*, 2003.

[17] 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.

[18] Hui Han, Hongyuan Zha, and C Lee Giles. Name disambiguation in author citations using a k-way spectral clustering method. In *Digital Libraries, 2005. JCDL'05. Proceedings of the 5th ACM/IEEE-CS Joint Conference on*, pages 334–343. IEEE, 2005.

[19] Jian Huang, Seyda Ertekin, and C Lee Giles. Fast author name disambiguation in citeseer. *ISI Technical Report*, 66, 2006.

[20] Matthew A Jaro. Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida. *Journal of the American Statistical Association*, 84(406):414–420, 1989.

[21] Matthew A Jaro. Probabilistic linkage of large public health data files. *Statistics in medicine*, 14(5-7):491–498, 1995.

[22] In-Su Kang, Seung-Hoon Na, Seungwoo Lee, Hanmin Jung, Pyung Kim, Won-Kyung Sung, and Jong-Hyeok Lee. On co-authorship for author disambiguation. *Information Processing & Management*, 45(1):84–97, 2009.

[23] Itshak Lapidot. Self-organizing-maps with bic for speaker clustering. Technical report, IDIAP, 2002.

[24] Michael Levin, Stefan Krawczyk, Steven Bethard, and Dan Jurafsky. Citation-based bootstrapping for large-scale author disambiguation. *Journal of the American Society for Information Science and Technology*, 63(5):1030–1047, 2012.

[25] MarkLogic. Best database for integrating data from silos — marklogic. `http://www.marklogic.com/`, 2017. (Accessed on July, 2017).

[26] MarkLogic. Task server configuration help — marklogic 9 product documentation. `https://docs.marklogic.com/admin-help/task-server`, July 2017. (Accessed on July, 2017).

[27] Staša Milojević. Accuracy of simple, initials-based methods for author name disambiguation. *Journal of Informetrics*, 7(4):767–773, 2013.

[28] Fakhri Momeni and Philipp Mayr. Evaluating co-authorship networks in author name disambiguation for common names. In *International Conference on Theory and Practice of Digital Libraries*, pages 386–391. Springer, 2016.

[29] Hien T Nguyen and Tru H Cao. Named entity disambiguation: A hybrid statistical and rule-based incremental approach. In *The Semantic Web*, pages 420–433. Springer, 2008.

[30] Byung-Won On, Dongwon Lee, Jaewoo Kang, and Prasenjit Mitra. Comparative study of name disambiguation problem using a scalable

blocking-based framework. In *Proceedings of the 5th ACM/IEEE-CS joint conference on Digital libraries*, pages 344–353. ACM, 2005.

[31] ORCID. Orcid — connectiong research and researchers. `https://orcid.org/`, 2017. (Accessed on January, 2017).

[32] Yanan Qian, Qinghua Zheng, Tetsuya Sakai, Junting Ye, and Jun Liu. Dynamic author name disambiguation for growing digital libraries. *Information Retrieval Journal*, 18(5):379–412, 2015.

[33] Jane Qiu. Scientific publishing: Identity crisis. *Nature News*, 451(7180): 766–767, 2008.

[34] Neil R Smalheiser and Vetle I Torvik. Author name disambiguation. *Annual review of information science and technology*, 43(1):1–43, 2009.

[35] Jie Tang, Alvis Cheuk M Fong, Bo Wang, and Jing Zhang. A unified probabilistic framework for name disambiguation in digital library. *Knowledge and Data Engineering, IEEE Transactions on*, 24(6):975–987, 2012.

[36] 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.

[37] 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.

[38] World Wide Web Consortium (W3C). Xquery 3.0: An xml query language. `https://www.w3.org/TR/xquery-30/`, July 2017. (Accessed on July, 2017).

[39] Henning Weiler. Authormagic: A concept for author disambiguation in large-scale digital libraries. 2012.

[40] Worldometers. World population clock. `http://www.worldometers.info/world-population/`, 2016. (Accessed on April, 2016).

[41] Kai-Hsiang Yang, Hsin-Tsung Peng, Jian-Yi Jiang, Hahn-Ming Lee, and Jan-Ming Ho. Author name disambiguation for citations using topic and web correlation. In *Research and advanced technology for digital libraries*, pages 185–196. Springer, 2008.