TOWARDS NAME DISAMBIGUATION: RELATIONAL, STREAMING, AND PRIVACY-PRESERVING TEXT DATA

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Baichuan Zhang

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

December 2017

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF DISSERTATION APPROVAL

Dr. Mohammad Al Hasan, Co-Chair
Department of Computer Science
Dr. Christopher W. Clifton, Co-Chair
Department of Computer Science
Dr. Dan Goldwasser
Department of Computer Science
Dr. Xia Ning
Department of Computer Science
Dr. Ninghui Li
Department of Computer Science

Approved by:

Dr. William J. Gorman
Thesis Form Head

To my parents, wife and all friends.

ACKNOWLEDGMENTS

First of all, I would like to take this opportunity to thank my PhD advisor Dr. Mohammad Al Hasan for his continued support during my PhD study. He introduced me to the research in information retrieval, text mining, graph analysis, and data privacy, especially for the topic of name disambiguation. He guided me how to think critically and independently about a research problem, how to design experiments to validate the research ideas, and how to write high quality papers to present research contributions. He always gave me freedom to investigate various research problems which match my interest and skill-set. I also learned a lot by taking his graduate level data mining and algorithm courses, which is beneficial to my research and full-time job hunting.

Next I would like to pay gratitude to all professors who served as my thesis committee members, namely Dr. Chris Clifton, Dr. Dan Goldwasser, Dr. Ninghui Li, and Dr. Xia Ning for their guidance, encouragement, and suggestions to improve my thesis work. Also I like to thank Dr. Sutanay Choudhury from Pacific Northwest National Lab (PNNL) for being a great mentor when I was a research intern at PNNL in summer 2015. He guided me through the area of knowledge graph construction and link prediction. A special note of appreciation goes to Yanbo Liang from Hortonworks Inc, who was my mentor when I was a big data engineer intern there in summer 2016. He introduced me to the distributed framework such as Apache Spark. My engineering skills improved a lot under his short-term guidance. I also would like to thank Dr. Zhonghua Qu for being a ideal mentor when I was a software engineering PhD intern in ads ranking team at Facebook Seattle in summer 2017. Additionally, a great appreciation goes to Dr. Pin-Yu Chen from IBM Thomas J. Watson Research Center. We collaborated several research papers together and I had much deeper understanding of linear algebra because of him during the fruitful collaboration.

I also would like to thank faculty members of Compute Science Department, Indiana University Purdue University Indianapolis (IUPUI) from whom I learned a lot in these four years. I appreciate Dr. Murat Dundar for contributing part of my thesis. Specifically, he introduced me the topic of Bayesian non-exhaustive learning and guided me to adapt this technique into online name disambiguation. Under his tremendous help, we published this work as a research track full paper in ACM International Conference on Information and Knowledge Management (CIKM 2016). I also like to thank him for teaching me machine learning course. Under his guidance, we published the course project, which is relevant to the topic of unsupervised feature learning for bacteria image classification, into IEEE International Conference on Machine Learning and Applications (ICMLA 2015). I would like to thank Dr. Xia Ning for teaching me recommender system course. Besides, she served in my oral qualifier exam committee, preliminary exam committee and final dissertation exam committee. I thank her for involvement in every step of my Ph.D. journey. I also like to thank Dr. Rajeev R. Raje for teaching me theory of programming languages. I am also specially grateful to the departmental staff, Nicole and Joan, for their continuous and whole hearted effort to make me feel easier in the department.

Finally, I am grateful to my parents and my wife for their long-term support and love. Their love and companionship have been my greatest motivation to complete my PhD degree. I would like to thank them for continuous patience and encouragement in these stressful times.

TABLE OF CONTENTS

Page

LIST OF TABLES x

LIST OF FIGURES xii

ABSTRACT xiv

1 INTRODUCTION 1

1.1 Contribution of This Dissertation 4

1.2 Organization of This Dissertation 7

2 RELATED WORK 8

2.1 Name Disambiguation 8

2.2 Neural Network Embedding 11

2.3 Non-Exhaustive Classification and Novel Class Discovery 12

2.4 Privacy-preserving Data Publishing 12

3 NAME DISAMBIGUATION FROM LINK DATA IN A ANONYMIZED COLLABORATION GRAPH 16

3.1 Introduction 16

3.2 Solution Overview 18

3.3 Methods 20

3.3.1 Obtaining Cluster Quality Score 21

3.3.1.1 Normalized Cut based Score 21

3.3.2 Obtaining Temporal Mobility Score 23

3.3.2.1 Temporal Mobility Score using KL Divergence 25

3.3.3 Linear Model using NC-score and TM-Score 26

3.4 Pseudo-code and Complexity Analysis 27

3.4.1 Supervised Classification Setup 28

3.5 Experiments and Results 29

Page

3.5.1 Evaluation of Unsupervised Disambiguation 31
3.5.2 Evaluation of Supervised Disambiguation 33
3.5.3 Comparison with Existing Works 35
3.5.4 Study of Parameter Sensitivity 36
3.5.5 Study of Dataset Bias 37
3.5.6 Study of Running Time 37
3.5.7 Real-life Case Study 38
3.6 Chapter Summary 39
4 NAME DISAMBIGUATION IN ANONYMIZED GRAPHS USING NETWORK EMBEDDING 41
4.1 Introduction 41
4.2 Problem Formulation 43
4.3 Method 47
4.3.1 Model Formulation 47
4.3.2 Model Optimization 50
4.3.3 Pseudo-code and Complexity Analysis 52
4.3.4 Mining Hard Negative Training Instance 53
4.4 Experiments and Results 54
4.4.1 Datasets 54
4.4.2 Competing Methods 56
4.4.3 Experimental Setting and Implementation 57
4.4.4 Comparison among Various Name Disambiguation Methods . . 59
4.4.5 Parameter Sensitivity of Embedding Dimension 61
4.4.6 Performance Comparison over the Number of Clusters 62
4.4.7 Component Contribution Analysis 63
4.4.8 Negative Instance Sampling Strategy Analysis 65
4.4.9 Convergence Analysis 65
4.5 Chapter Summary 66

Page

5 BAYESIAN NON-EXHAUSTIVE CLASSIFICATION FOR ACTIVE ONLINE NAME DISAMBIGUATION 67
5.1 Introduction 67
5.2 Online Name Disambiguation Challenges 71
5.3 Online Name Disambiguation on Bibliographic Data 73
5.3.1 Feature Matrix Construction and Preprocessing 74
5.3.2 Problem Formulation 75
5.4 Methodology 76
5.4.1 Dirichlet Process Gaussian Mixture Model 76
5.4.2 Online Inference by One-Pass Gibbs Sampler 79
5.4.3 Online Inference by Particle Filtering 80
5.5 Active Online Name Disambiguation 84
5.6 Experimental Results 84
5.6.1 Datasets 85
5.6.2 Competing Methods 86
5.6.3 Experimental Setting and Implementation 87
5.6.4 Performance Comparison with Competing Methods 88
5.6.5 Feature Contribution Analysis 92
5.6.6 Study of Running Time 93
5.6.7 Study of Parameter Sensitivity 94
5.6.8 Performance over the Number of Observed Online Records . . . 95
5.6.9 Results of Active Online Name Disambiguation 96
5.7 Chapter Summary 97
6 FEATURE SELECTION FOR CLASSIFICATION UNDER ANONYMITY CONSTRAINT 98
6.1 Objective and Motivation 98
6.2 Introduction 98
6.2.1 Our Contributions 103
6.3 Privacy Basics 105

Page

6.4 Problem Statement 107
6.5 Methods 111
6.5.1 Maximal Itemset Based Approach 112
6.5.1.1 Maximal Feasible Feature Set Generation 114
6.5.1.2 Classification Utility Function 114
6.5.1.3 Maximal Itemset Based Method (Pseudo-code) . . . 116
6.5.2 Greedy with Modular and Sub-Modular Objective Functions . 117
6.5.2.1 Submodularity, and Modularity 118
6.5.2.2 Greedy Method (Pseudo-code) 120
6.6 Experiments and Results 121
6.6.1 Privacy Preserving Classification Tasks 122
6.6.2 Experimental Setting 123
6.6.3 Name Disambiguation 126
6.6.4 Spam Email Filtering 128
6.7 Chapter Summary 130
7 FUTURE WORK AND CONCLUSION 131
REFERENCES 133
VITA 143

LIST OF TABLES

Table Page

1.1 Various aspects of name disambiguation algorithms presented in this dissertation 7

3.1 Comparison between our method and [37] using classification accuracy (%) on 10-fold cross-validation 32

3.2 Comparison between our method and [37] using AUC on 10-fold cross-validation 32

3.3 Precision of multi-node class @ top-k(%) for DBLP dataset 33

3.4 Precision of multi-node class @ top-k(%) for Arnetminer dataset 33

3.5 Running time result in DBLP 38

3.6 Real-life case study showing prominent researchers in DBLP and Arnetminer datasets. The bold values correspond to the cases for which the prediction of our method is wrong 38

4.1 Arnetminer name disambiguation dataset 55

4.2 CiteSeerX name disambiguation dataset 55

4.3 Comparison of Macro-F1 values between our proposed method and other competing methods for name disambiguation task in Arnetminer dataset (embedding dimension = 20). Paired t-test is conducted on all performance comparisons and it shows that all improvements are significant at the 0.05 level. 58

4.4 Comparison of Macro-F1 values between our proposed method and other competing methods for name disambiguation task in CiteSeerX dataset (embedding dimension = 20). Paired t-test is conducted on all performance comparisons and it shows that all improvements are significant at the 0.05 level. 59

4.5 Negative instance sampling strategy comparison in our proposed network embedding model for name disambiguation task (embedding dimension = 20). The Macro-F1 results are averaged out over 10 name references in each of the datasets. 64

5.1 Arnetminer name disambiguation dataset 85

Table Page

5.2 Comparison of mean-F1 values using records with most recent 2 years as test set. Paired t-test is conducted on all performance comparisons and it shows that all improvements are significant at the 0.05 level. 89

5.3 Comparison of mean-F1 values using records with most recent 3 years as test set. Paired t-test is conducted on all performance comparisons and it shows that all improvements are significant at the 0.05 level 89

5.4 Results of number of distinct real-life persons under our proposed Bayesian non-exhaustive classification framework using most recent 3 years' records as test set 92

6.1 A toy 2-class dataset with binary feature-set 101

6.2 Projections of the dataset in table 6.1 on two feature-sets (Feature Set-1 and Feature Set-2) 101

6.3 Statistics of real-world datasets 121

6.4 AUC comparison among different privacy methods for the name disambiguation task 126

6.5 Comparison among different privacy methods for Email dataset using AUC 128

LIST OF FIGURES

Figure Page

3.1 A toy example of clustering based name entity disambiguation 19

3.2 Temporal mobility example 24

3.3 Unsupervised disambiguation experimental results: (a) on Arnetminer and (b) on DBLP 30

3.4 Parameter sensitivity of our method on two datasets 36

3.5 Bias effect in DBLP and Arnetminer 37

4.1 Paper count distribution of name reference "Lei Wang" 43

4.2 The effects of embedding dimension on the name disambiguation performance 61

4.3 Macro-F1 results of multiple L values on name reference "Lei Wang" using our method, GF, and LINE (embedding dimension = 20). 62

4.4 Component contribution analysis in terms of name disambiguation performance using Arnetminer and CiteSeerX as a whole source (embedding dimension = 20). 63

4.5 Convergence analysis in terms of both objective loss and Macro-F1 of name reference "Lei Wang" using our proposed network embedding model for name disambiguation (embedding dimension = 20). 64

5.1 Name ambiguity evolution for the name "Jing Zhang" 68

5.2 Bayesian non-exhaustive classification framework for active online name disambiguation 69

5.3 Feature contribution analysis using most recent 2 years' publication records as test set. The results are averaged out over all 25 name references for better visualization. 93

5.4 The effects of latent dimension h, concentration parameter α in Dirichlet process, and number of particles M in particle filtering on the online name disambiguation performance using most recent 2 years' records as test set. The results are averaged out over all 25 name references. 94

Figure Page

5.5 Online name disambiguation performance over the number of observed online records on name reference "Jing Zhang" using most recent 3 years' records as test set. 95

5.6 Mean-F1 comparison between our proposed active selection and random selection with respect to different ratios of queried online records on name reference "Jing Zhang" using most recent 3 years' records as test set. The higher the curve, the better the performance. 96

6.1 Classification performance of differential privacy based methods for different ϵ values on three datasets. Laplace mechanism is on the left and exponential mechanism is on the right. Each group of bars belong to one specific dataset, and within a group different bars represent different ϵ values.129

ABSTRACT

Baichuan Zhang Ph.D., Purdue University, December 2017. Towards Name Disambiguation: Relational, Streaming, and Privacy-Preserving Text Data. Major Professor: Mohammad Al Hasan.

In the real world, our DNA is unique but many people share names. This phenomenon often causes erroneous aggregation of documents of multiple persons who are namesake of one another. Such mistakes deteriorate the performance of document retrieval, web search, and more seriously, cause improper attribution of credit or blame in digital forensics. To resolve this issue, the name disambiguation task [1] is designed to partition the documents associated with a name reference such that each partition contains documents pertaining to a unique real-life person. Existing algorithms for this task mainly suffer from the following drawbacks. First, the majority of existing solutions substantially rely on feature engineering, such as biographical feature extraction, or construction of auxiliary features from Wikipedia. However, for many scenarios, such features may be costly to obtain or unavailable in privacy sensitive domains. Instead we solve the name disambiguation task in restricted setting by only leveraging the relational data in the form of anonymized graphs. Second, most of the existing works for this task operate in a batch mode, where all records to be disambiguated are initially available to the algorithm. However, more realistic settings require that the name disambiguation task should be performed in an online streaming fashion in order to identify records of new ambiguous entities having no preexisting records. Finally, we investigate the potential disclosure risk of textual features used in name disambiguation and propose several algorithms to tackle the task in a privacy-aware scenario. In summary, in this dissertation, we present a number

[1] In this dissertation, we use the terms "name disambiguation", "entity disambiguation", "name entity disambiguation", and "author name disambiguation" interchangeably.

of novel approaches to address name disambiguation tasks from above three aspects independently, namely relational, streaming, and privacy preserving textual data.

In the first framework, we propose a method for solving the name disambiguation task from relational data obtained from a collaboration network. Instead of using attributes pertaining to a real-life person, our proposed methodology leverages only relational data in the form of anonymized graphs. Then the method returns a real-valued score to rank the vertices of a network based on their likelihood for being an ambiguous node. Furthermore, in order to perform the actual disambiguation, we present a novel network embedding model to embed each document in a low dimensional vector space where the name disambiguation can be solved by a hierarchical agglomerative clustering algorithm. Our experimental results demonstrate that the proposed method is significantly better than the existing name disambiguation methods working in the similar setup.

In the second framework, we present a Bayesian non-exhaustive classification framework for solving online name disambiguation. In particular, we present a Dirichlet Process Gaussian Mixture Model (DPGMM) as a core engine for the online name disambiguation task. Meanwhile, two online inference algorithms, namely one-pass Gibbs sampler and Sequential Importance Sampling with Resampling (also known as particle filtering), are proposed to simultaneously perform online classification and new class discovery. As a case study, we consider bibliographic data in a temporal stream format and disambiguate authors by partitioning their papers into homogeneous groups. We also propose an interactive version of our online name disambiguation method to leverage user feedback to improve prediction accuracy.

Finally, we investigate the potential disclosure risk of textual features used in name disambiguation [2]. The goal of this work is to select a subset of textual features such that on the reduced feature set, the data has small disclosure risk, and the utility of data is maximally preserved for performing the downstream name disambiguation task. Furthermore, we design an anonymization metric, called k-anonymity by con-

[2]According to the literature [1], a list of key-phrase based textual features sustain high potential risk to reveal personal identity.

tainment, to measure the privacy risk of selected textual features. The experimental results demonstrate that our proposed anonymization protocol performs better to maintain the data utility compared to several state-of-art privacy metrics, such as k-anonymity and differential privacy.

1. INTRODUCTION

Popular names are shared by many people around the world. When such names are mentioned in various on-line and off-line documents, more often, ambiguity arises; i.e., we cannot easily deduce from the document context which real-life person a given mention is being referred to. Being unable to resolve this ambiguity often leads to erroneous aggregation of documents of multiple persons who are namesake of one another. Such mistakes deteriorate the performance of document retrieval, web search, and bibliographic data analysis. In web search, name disambiguation is important for sanitizing search results of ambiguous queries. For example, an online search query for "Michael Jordan" may retrieve pages of former US basketball player, the pages of UC Berkeley machine learning professor, the pages of former CEO of General Electric (GE), and the pages of other persons having that name, and name disambiguation is needed to organize those pages in homogeneous groups. For bibliometrics and library sciences, many distinct authors in the academic world share the same name. As a result, the bibliographic servers that maintain publication data may mistakenly aggregate the articles from multiple scholars (sharing the same name) into a unique profile in some digital repositories. For an example, the Google scholar profile associated with the name "Yang Chen" (GS) [1] is verified as the profile page of a Computer Graphics PhD candidate at Purdue University, but based on our labeling, more than 20 distinct persons' publications are mixed under that profile mistakenly. Such issues in library science over- or under-estimate a researcher's citation related impact metrics. Beyond the academic world, name ambiguity conundrum also causes misidentification during counter-terrorism efforts, leading to severe distress to many individuals who happen to share names with wanted suspects.

[1] https://scholar.google.com/citations?user=gl26ACAAAAAJ&hl=en

The *name disambiguation* task is used to resolve the name ambiguity problem. Formally speaking, given a large collection of records pertaining to a single name value, the name disambiguation task partitions the records into groups of records, such that each group belongs to a unique real-life person. In the above definition the term "record" refers to any form of collective information associated with the mention of a given name. For instance, in a digital repository of academic publications, a record is simply the citation context (title, co-authors, and venue) of a paper. In case of a mention of a name in an online news article, a record may include information such as article title, sentence context of the mention, and other associated name references within the article. For a social network profile, the record may contain publicly available friend-list, and text from the posts in that profile.

Due to its importance, the name disambiguation task has attracted substantial attention from information retrieval and data mining communities. However, the majority of existing solutions [2–6] for this task use biographical features such as name, address, institutional affiliation, email address, and homepage. Also, contextual features such as collaborator, community affiliation, and external data source such as Wikipedia are used in some works [6, 7]. Using the biographical features is acceptable for disambiguation of authors in bibliometrics domain, but in many scenarios, for example in the national security related applications, biographical features are hard to obtain, or they may even be illegal to obtain unless a security analyst has the appropriate level of security clearance. Besides, in real-world social networks (e.g., Twitter, Facebook, and LinkedIn), some users may choose a strict privacy setting that restricts the visibility of their profile information and posts. For such resource bounded scenarios, many existing name disambiguation techniques [5, 6, 8–10], which compute document similarity using biographical attributes are not applicable. Instead, we solve the name disambiguation task by only leveraging the relational data in the form of anonymized graphs. Specifically, relational data under our context refers to the co-occurrence of two entities in the graphs, and biographical attributes of each node in the graph are unavailable in such settings. In this dissertation, de-

signing an effective name disambiguation solution only utilizing relational data is one of our major focuses.

The majority of the existing approaches to name disambiguation in literature operate in a batch mode, where all the records to be disambiguated are assumed to be accessible to the algorithm initially. This assumption requires running a new disambiguation task every time a record is added to the collection. However, due to the fast growth of digital libraries, or streaming data sources (Twitter, Facebook), rerunning disambiguation process on the whole data every time a new record is added would not be very economical. Instead, it is more practical to perform this task in an incremental fashion by considering the streaming nature of records. Designing an incremental, i.e., online, name disambiguation is challenging as the method must be able to adapt to a *non-exhaustive* training dataset [2]. In other words, it should be able to identify records belonging to new ambiguous persons who do not have any historical records in the system. After identification, the learning algorithm must re-configure the model (for instance, update the number of classes, k) so that it can correctly recover future records of this newly found ambiguous person. This is an important requirement because in real-life, for a common name, a significant number of streaming records belongs to novel (not yet seen) persons sharing that name. Besides non-exhaustiveness, *online verification* is another desirable property for an incremental name disambiguation system. Such a system asks users to provide feedback on the correctness of its prediction. Feedback collection can be automated by using online social networks or crowdsourcing platforms. As an example, consider the online digital library platform ResearchGate [3]; it performs author name disambiguation by asking a potential researcher whether he is the author of a paper before adding that paper to that person's profile. Human feedback significantly improves the accuracy of a name disambiguation task; however, to reduce human effort the system should consult the human as infrequently as possible, and the consultation should be made

[2] A training dataset is called exhaustive if it contains records for all values (classes) of the target variable, otherwise it is called non-exhaustive.

[3] `https://www.researchgate.net/`

for documents, for which the human feedback would yield the maximum utility for reconfiguring the model. Thus, designing an active online name disambiguation system that can accommodate streaming non-exhaustive data is another focus of this dissertation.

The majority of works have demonstrated that textual features are one of most powerful signals to solve name disambiguation. Such an exercise is quite common in homeland security for disambiguating multiple suspects from their digital footprints. However, privacy is important in such applications, as a list of key-phrase based features can potentially reveal personal identity. The objective of this work is to solve the feature selection problem such that the selected textual attributes satisfy non-disclosure requirements, at the same time they achieve high name disambiguation performance as well. In addition, we present a privacy metric to assess how much privacy is preserved for each individual. Therefore, solving name disambiguation in a privacy-preserving manner is another focus of this dissertation.

This dissertation mainly focuses on three aspects of name disambiguation task, which is relational, streaming, and privacy-preserving textual data. Note that we do not build a single system for performing all these three aspects in a unified manner, instead we address each of them independently. We summarize our contribution of this dissertation in the following subsection.

1.1 Contribution of This Dissertation

We summarize the major contributions of this dissertation as below:

1. We propose a novel name disambiguation method. Instead of using attributes pertaining to a real-life person, our method leverages only relational data in the form of anonymized graphs. The proposed approach of using anonymous graphs works in the scenario where node attributes are not accessible. Such a method is very useful in sensitive data analysis, such as surveillance over social network data for anomaly detection. Specifically, the proposed name

disambiguation solution consists of two phases: ambiguity identification and actual disambiguation. In the ambiguity identification step, we consider an anonymized social network. Each node in this network corresponds to a reference to a named entity, and each edge corresponds to collaboration among different named entities. The edges are labeled with time-stamps representing the time when a collaboration took place. Our solution uses the timestamped network topology around a vertex of the network and by using an unsupervised method it produces a real-valued score for that vertex. This score represents the degree to which a given anonymized reference (a vertex) is pure. The smaller the score, the more likely that the reference may comprise of records of multiple real-life entities. For a given vertex, the method provides the desired score in a few seconds, so one can always use it as a pre-filter to identify a small set of target nodes for which more thorough analysis can be made subsequently. The work of name ambiguity identification in anonymized graphs is published in the proceedings of Advances in Social Network Analysis and Mining (ASONAM) 2014 as a research track short paper [11], and the extended version is published in Social Network Analysis and Mining (SNAM) [12]. Furthermore, if ambiguity of the given name entity is detected, in actual disambiguation stage, we utilize a novel representation learning model to embed each of its corresponding document in a low dimensional vector space where actual disambiguation can be solved by a hierarchical agglomerative clustering algorithm. The work of actual disambiguation using network representation learning is published in the proceedings of the Conference of Information and Knowledge Management (CIKM) 2017 as a research track full paper [13].

2. We propose a Bayesian non-exhaustive classification framework for solving online name disambiguation. The proposed algorithm can not only classify records of known persons represented in the training data by their existing records, but can also identify records of new ambiguous persons with no existing records included in the initial training dataset. In particular, we present a Dirichlet Pro-

cess Gaussian Mixture Model (DPGMM) as a core engine for the online name disambiguation task. Meanwhile, two online inference algorithms, namely one-pass Gibbs sampler and Sequential Importance Sampling with Resampling (also known as particle filtering), are proposed to simultaneously perform online classification and new class discovery. As a case study we consider bibliographic data in a temporal stream format and disambiguate authors by partitioning their papers into homogeneous groups. Our experimental results demonstrate that the proposed method is significantly better than existing methods for performing online name disambiguation task. We also propose an interactive version of our online name disambiguation method designed to leverage user feedback to improve prediction accuracy. The work is published in the proceedings of CIKM 2016 as a research track full paper [14]. The extended version is publicly available in arXiv [15].

3. Finally, this dissertation also investigates the potential disclosure risk of textual features for the task of name disambiguation. The objective of this work is to solve the feature selection problem such that the selected textual attributes satisfy non-disclosure requirements, at the same time they also achieve high name disambiguation performance. In particular, inspired by the privacy-preserving data publishing (PPDP) research [1, 16–19], we design an anonymization metric, named k-anonymity by containment, to quantify the privacy protection of selected textual features. The experimental results demonstrate that our proposed anonymization protocol performs better to maintain the data utility in terms of name disambiguation task compared to several the-state-of-art privacy metrics, such as k-anonymity and differential privacy. For this work, it is published in the Transactions on Data Privacy (TDP) [20].

Table 1.1.: Various aspects of name disambiguation algorithms presented in this dissertation

	Privacy Preserving	Relational Data	Textual Data	Streaming Data	Anonymous Graph	Disambiguate Author	Disambiguate Document	User Feedback
Chapter 3	✓	✓			✓	✓		
Chapter 4	✓	✓			✓		✓	
Chapter 5				✓			✓	✓
Chapter 6	✓		✓			✓		

1.2 Organization of This Dissertation

The organization of this dissertation is illustrated as follows. In Chapter 2, we discuss related works in the topics of name disambiguation and network embedding. We also discuss privacy-preserving data publishing, privacy-aware feature selection and non-exhaustive classification. In both Chapter 3 and Chapter 4, we discuss name disambiguation using relational data. Specifically, name ambiguity identification in anonymized graphs is presented in Chapter 3. The discussion is continued in Chapter 4 for actual disambiguation using network representation learning technique. The Bayesian non-exhaustive classification for active online name disambiguation is presented in Chapter 5, which is relevant to name disambiguation using streaming data. The investigation of feature selection for name disambiguation under anonymity protection is described in Chapter 6, which is linked to the name disambiguation leveraging privacy-preserving textual data. Finally future works and conclusion are discussed in Chapter 7. In order to better understand the topics presented in different chapters, we summarize different aspects of name disambiguation algorithms shown in Table 1.1.

2. RELATED WORK

In this chapter, we discuss the related work into the following four categories, namely name entity disambiguation, neural network embedding, non-exhaustive learning for the application of novel class discovery, and privacy-preserving data publishing.

2.1 Name Disambiguation

In existing works, name disambiguation task is studied for various entities; examples include disambiguation on Encyclopedic knowledge or Wikipedia Data [2, 4, 5, 7, 21, 22], citation data [10, 23–28], email data [29], and text documents [6, 30–32].

In terms of methodologies, both supervised [2, 10] and unsupervised [23] approaches are considered. For supervised method, a distinct entity can be considered as a class, and the objective is to classify each document to one of the classes. Han et al. [23] use such a framework, and propose two supervised methods, one using a generative model, and the other using SVM. In another supervised approach, Bunescu et al. [2] solve name disambiguation by designing and training a disambiguation SVM kernel that exploits the high coverage and rich structure of the knowledge encoded in an online encyclopedia. However, the main drawback of supervised methods is the unavailability of labeled data for training.

For unsupervised name disambiguation, the collaboration events are partitioned into several clusters with a goal that each cluster contains the events corresponding to a unique entity. Han et al. [23] propose one of the earliest unsupervised name disambiguation methods, which is based on K-way spectral clustering. They apply their method for name disambiguation in an academic citation network. For each name dataset, they calculate a Gram matrix representing similarities between different citations and apply K-way spectral clustering algorithm on the Gram matrix to obtain

the desired clusters of the citations. In another unsupervised approach, Cen et al. [3] compute pairwise similarity for publication events that share the same author name string (ANS) and then use a novel hierarchical agglomerative clustering with adaptive stopping criterion (HACASC) to partition the publications into different author clusters. Malin [33] proposes another cluster-based method that uses social network structure.

Probabilistic relational models, specifically graphical models have also been used for solving the name disambiguation task. For example, the authors in [28] propose a constraint-based probabilistic model for semi-supervised name disambiguation using hidden Markov random fields (HMRF). They define six types of constraints and employ the EM algorithm to learn the HRMF model parameters. In another work, Tang et al. [8, 9] present two name disambiguation methods that are based on a pairwise factor graph model. They target name disambiguation in academic datasets. In their work, the authorship of a paper is modeled as edges between observation variables (papers) and hidden variables (author labels). Features of each paper and relationships, such as co-publication-venue and co-author, have impact on the probability of each assignment of labels. The similarity between two clusters is encoded in different factors (edge potentials) on different features. The clustering process iterates over different author label assignments and selects the one with maximal probability. LDA based context-aware topic models has also been used for name disambiguation [30].

To build the features for classification, clustering, or probabilistic models, most of the existing works use biographical and contextual attributes of the entities or external sources such as Wikipedia, web search results and online encyclopedia. In almost every work on name disambiguation, person's name, email address, and institutional affiliation are used. It is not surprising, because biographical features are highly effective for name disambiguation. For instance, a set of recent works [34, 35] report around 99% accuracy on a data mining challenge dataset prepared by Microsoft research. However, the attempt of extracting biographical or external data sustains the risk of privacy violation. To address this issue, a few works [33, 36, 37]

have considered name disambiguation using anonymized graphs without leveraging the node attributes. The central idea of this type of works is to exploit graph topological features to solve the name disambiguation problem without intruding user privacy through the collection of bibliographical attributes. For example, authors in [37] characterized the similarity between two nodes based on their local neighborhood structures using graph kernels and solved the name disambiguation problem using SVM. However, the major drawback of the proposed method in [37] is that it can only detect entities that should be disambiguated, but fails to further partition the documents into their corresponding homogeneous groups. In addition to that, the authors in [33, 36] formulate the name disambiguation as a graph problem and utilize a random walk based approach to tackle the disambiguation task. However, both works suffer from a similar issue to that described above.

Most existing methods tackle disambiguation task in a batch setting, where all records to be resolved are initially available to the algorithm, which makes these techniques unsuitable for disambiguating a future record. In recent years, online name disambiguation was considered in a few works [38–41]. These techniques perform name disambiguation incrementally without the need to retrain the system every time a new record is received. Specifically, Khabsa et al. [39] use an online variant of DBSCAN, a well-known density-based clustering technique, to cluster new records incrementally as they become available. Since DBSCAN does not use a fixed number of clusters it can adapt to the non-exhaustive scenario by simply assigning a new record to a new cluster, as needed. However, DBSCAN is quite susceptible to the choice of parameter values, and depending on the specific values chosen, a record of an emerging class can be simply labeled as an outlier instance. [38] proposes a two stage framework for online name disambiguation. The first stage performs batch name disambiguation to disambiguate all the records that appeared no later than a given time threshold using hierarchical agglomerative clustering. The second stage performs incremental name disambiguation to determine the class membership of a newly added record. However, the method uses a heuristic threshold to decide on the

cluster assignments of new records which makes the performance of this approach very sensitive to the choice of threshold parameter. [40] introduces an association rule based approach for detecting unseen authors in test set. The major drawback of their proposed solution is that it can only identify records of emerging authors in a binary setting but fails to further distinguish among them. Besides, the approach is not very robust with respect to the threshold parameter used in the association rule discovery. In addition to that, the authors in [42] present the Expectation Maximization based approach for the name disambiguation in Twitter streaming data. The authors in [43] propose a threshold based method to discover emerging entities with ambiguous names in the domain of knowledge base.

Another line of work approaches name disambiguation from an active learning perspective [9,44–46]. For example, the authors in [44] propose a method that queries the label information for the most ambiguous records. The authors in [9] present a pairwise factor graph model for active name disambiguation, which maximizes the utility of user's corrections for improving the disambiguation performance. Another recent work uses crowdsourcing for active name disambiguation [45]. However, all of these active name disambiguation techniques are proposed to tackle the offline setting. Finally, a survey article is also available, which presents a taxonomy of various name disambiguation methods in the existing literature [47].

2.2 Neural Network Embedding

For solving name disambiguation in an anonymized graph (details in chapter 4), our proposed solution utilizes a neural network embedding based approach [48–61]—a rather recent development in machine learning. Many of these methods are inspired by the word2vec based language model [62, 63]. Different from traditional graph embedding methods, such as Structure Preserving Embedding (SPE) [64], Local Linear Embedding (LLE) [65], and Laplacian Eigenmaps [66], the recently proposed network embedding methods, such as DeepWalk [49], LINE [51], PTE [50], and Node2Vec [52],

are more scalable and have shown better performance in node classification and link prediction tasks. Among these works, LINE [51] finds embedding of documents by using document-document similarity matrix, whereas in our work shown in chapter 4, we use multiple networks and perform a joint learning. PTE [50] performs a joint learning of multiple input graphs, but PTE needs labeled data. Finally, the embedding formulation and optimization of our proposed method are different than LINE or PTE. Specifically, we use a ranking based loss function as our objective function whereas mostly all the existing methods use K-L divergence based objective function.

2.3 Non-Exhaustive Classification and Novel Class Discovery

For solving the active online name disambiguation (details in chapter 5), our proposed solution utilizes Bayesian non-exhaustive classification technique. [67–71] are some of the existing works related to non-exhaustive learning and novel class discovery. Specifically, Akova et al. [67] propose a Bayesian approach for detecting emerging classes based on posterior probabilities. However, the decision function for identifying emerging classes uses a heuristic threshold and does not consider a prior model over class parameters; hence the emerging class detection procedure of this model is purely data-driven. Miller et al. [69] present a mixture model using expectation maximization (EM) for online class discovery. [68] proposes a sampling based online inference approach for emerging class discovery and the work is motivated by a bio-detection application. Additionally, the authors in [70, 71] present ensemble based classification methods for novel class detection in concept-drift data streams.

2.4 Privacy-preserving Data Publishing

In terms of privacy model, several privacy metrics have been widely used in order to quantify the privacy risk of published data instances, such as k-anonymity [72], t-closeness [73], ℓ-diversity [74], and differential privacy [75]. Existing works on privacy preserving data mining solve a specific data mining problem given a privacy constraint

over the data instances, such as classification [17], regression [76], clustering [16] and frequent pattern mining [77]. However, the solutions proposed in these works are strongly influenced by the specific data mining task and also by the specific privacy model. In fact, the majority of the above works consider distributed privacy where the dataset is partitioned among multiple participants owning different portions of the data, and the goal is to mine shared insights over the global data without compromising the privacy of the local portions. A few other works [78, 79] consider output privacy by ensuring that the output of a data mining task does not reveal sensitive information.

The k-anonymity privacy metric, due to its simplicity and flexibility, has been studied extensively over the years. The authors in [80] present the k-anonymity patterns for the application of association rule mining. Samarati [81] proposes formal methods of k-anonymity using suppression and generalization techniques. She also introduced the concept of minimal generalization. Meyerson et al. [82] prove that two definitions of k-optimality are $\mathcal{NP}$-hard: first, to find the minimum number of cell values that need to be suppressed; second, to find the minimum number of attributes that need to be suppressed. Henceforth, a large number of works have explored the approximation of anonymization [82, 83]. However, none of these works consider the utility of the dataset along with the privacy requirements. Kifer et al. [84] propose methods that inject utility in the form of data distribution information into k-anonymous and ℓ-diverse tables. However, the above work does not consider a classification dataset. Iyengar [85] proposes a utility metric called CM which is explicitly designed for a classification dataset. However, It assigns a generalization penalty over the rows of the dataset, and its performance is poor as we have shown in this work.

In recent years differential privacy [86–88] has attracted much attention in privacy research literatures. The authors in [89] propose a sampling based method for releasing high dimensional data with differential privacy guarantees. [90] proposes a method of publishing differential private low order marginals for high dimensional

data. Even though the authors in [89, 90] claim that they deal with high dimensional data, the dimensionality of data is at most 60 from the experiments in their works. [91] makes use of k-anonymity to enhance data utility in differential privacy. An interesting observation of this work is that differential privacy based method, by itself, is not a good privacy mechanism, in regards to maintaining data utility. [92] proposes a probabilistic top-down partitioning algorithm to publish set-valued data via differential privacy. The authors of [93] propose to utilize the exponential mechanism to release a decision tree based classifier that satisfies ϵ-differential privacy. However, in their work, privacy is embedded in the data mining process, hence they are not suitable as a data release mechanism, and more importantly they can only be used along with the specific classification model within which the privacy mechanism is built-in.

Our proposed solution for privacy-aware name disambiguation (details in chapter 6) is also relevant to the topic of privacy-aware feature selection. Empirical study for the use of feature selection in Privacy Preserving Data Publishing has been proposed in [94] [18]. However, in their work, they use feature selection as an add-on tool prior to data anonymization and do not consider privacy during the feature selection process. For our work, we consider privacy-aware feature selection with a twin objective of privacy preservation and utility maintenance. To the best of our knowledge, the most similar works to ours for the use of feature selection in privacy preserving data publishing are presented in [19, 95] recently. [19] considers privacy as a cost metric in a dynamic feature selection process and proposes a greedy based iterative approach for solving the task, where the data releaser requests information about one feature at a time until a predefined privacy budget is exhausted. However the entropy based privacy metric presented in this work is strongly influenced by the specific classifier. [95] presents a genetic approach for achieving k-anonymity by partitioning the original dataset into several projections such that each one of them adheres to k-anonymity. But the proposed method does not provide optimality guaranty. Finally,

a survey paper [96] presents various privacy preserving data publishing algorithms in the existing literature.

3. NAME DISAMBIGUATION FROM LINK DATA IN A ANONYMIZED COLLABORATION GRAPH

3.1 Introduction

On January 17, 2014, in his speech regarding the usage of phone surveillance data by NSA (National Security Agency), the USA President Barack Obama said, "This program does not involve the content of phone calls, or the names of people making calls. Instead, it provides a record of phone numbers and the times and lengths of calls—metadata that can be queried if and when we have a reasonable suspicion that a particular number is linked to a terrorist organization." In this talk he also mentioned the importance of balancing security and privacy in all surveillance works of government agencies. However, making this balance is not an easy task; respecting privacy does not allow tapping into someone's non-public biographical records; on the other hand, constrained analysis without detailed biographical data leads to numerous false identification and entity mixup. In this work, we are concerned with solving the task of name disambiguation without using biographical information—the input to our solution is link data collected from anonymized collaboration networks, similar to the one that Mr. Obama has explained.

Many research works are proposed for solving named entity disambiguation [2, 10, 21, 23, 29]. Existing works mostly use biographical features, such as name, address, institutional affiliation, email address, and Internet homepage; contextual features, such as coauthors/collaborators, and research keywords; and external data such as Wikipedia [21]. From methodological point of view, some of the works follow a supervised learning approach [10,37], while others use unsupervised clustering [3,23,33,97]. There exist quite a few solutions that use graphical models [8,9,28,98]. What is common among all these works is that they use many biographical features including

name, and affiliation, so they cannot protect the privacy of the actors in the dataset. Using biographical features is acceptable for entity disambiguation of authors in the field of bibliometrics, but in many scenarios, for example in the national security related applications, biographical features are hard to obtain, or they may even be illegal to obtain unless a security analyst has the appropriate level of security clearance. Besides, in real-world social networks (e.g., Twitter, Facebook, and LinkedIn), some users may choose a strict privacy setting that restricts the visibility of their profile information and posts. For such privacy-preserving applications, it is more desirable to use a fast indicator that identifies a small list of suspicious data references for which biographical data can be queried after the approval of a privacy management officer.

In this work, we consider an anonymized social network. Each node in this network corresponds to a reference to a named entity, and each edge corresponds to a collaboration event among different named entities. The edges are labeled with timestamps representing the time when a collaboration took place. As we have discussed earlier, we can think of such a network as the anonymized email/communication network that the NSA uses to identify suspects. Our solution to entity disambiguation in an anonymized network uses the timestamped network topology around a vertex of the network and by using an unsupervised method it produces a real-valued score for that vertex. This score represents the degree to which a given anonymized reference (a vertex) is pure. The smaller the score, the more likely that the reference may comprise of records of multiple real-life entities. For a given vertex, the method provides the desired score in a few seconds, so one can always use it as a pre-filter to identify a small set of target nodes for which more thorough analysis can be made subsequently. Alternative to an unsupervised approach, our method can also be adapted to a supervised classification system for predicting the purity status of a node, when a labeled dataset is available.

We claim the following contributions in this work:

- We design the task of solving name entity disambiguation using only graph topological information. This work is motivated by the growing need of data analysis without violating the privacy of the actors in a social network.
- We propose a simple solution that is robust and it takes only a few seconds to disambiguate a given node in real-life academic collaboration networks. The proposed method returns a real-valued score to rank the vertices of a network based on their likelihood for being an ambiguous node. So the score can be used as a pre-filter for identifying a small set of ambiguous references for subsequent analysis with a full set of features. Besides, the score can also be used independently as a feature for classification based solutions for name disambiguation.
- We use two real-life datasets for evaluating the performance of our solution. The results show that the method performs satisfactorily, considering the fact that it uses only the topology of a node in its analysis.

3.2 Solution Overview

We assume that a collaboration network $G(V, E)$ is given, where each node $u \in V$ represents an entity reference which in real-life may be linked to multiple persons. For every edge $e \in E$ we are also provided with a list $T(e)$ which represents the discrete time-point at which the collaboration events between the corresponding nodes have taken place. Our objective is to predict how likely it is that the node u is a multi-node, i.e., it comprises of collaboration records of multiple persons. We use a linear model to produce a numeric score $s(u)$, which represents the likelihood that u is a pure node.

To solve the problem in the setup discussed in the last paragraph, we first construct the ego network of u, $G_u \subseteq G$, which is an induced subgraph of G consisting of ego node u and all of its direct neighbors (these nodes are called "alters"). Since G_u is

an induced subgraph, it preserves the ties between u and the alters and also the ties between a pair of alters. We hypothesize that if u is a multi-node, the graph G_u will form many disjoint clusters, once the node u and all of its incident edges from G_u are removed; each of these clusters corresponds to one of the many real-life entities that have been merged together under the reference u. This hypothesis is built from the transitivity property of a social network, which states that the friends of your friend have high likelihood to be friends themselves [99]. Thus, if v and w are friends of u, with high likelihood, there are edges between v and w. However, when u is a multi-node corresponding to k different people, the friends of u are partitioned into at least k disjoint clusters.

In Figure 3.1, we illustrate our hypothesis. Assume that the triangle shaped node is u, and the graph in this figure corresponds to the ego network of u, G_u. We also assume that u is a multi-node consisting of two named entities. So the removal of the node u (along with all of its incident edges) from G_u makes two disjoint clusters; this phenomenon is illustrated in the lower part of figure 3.1. The vertices of these clusters are shown using circles and squares respectively.

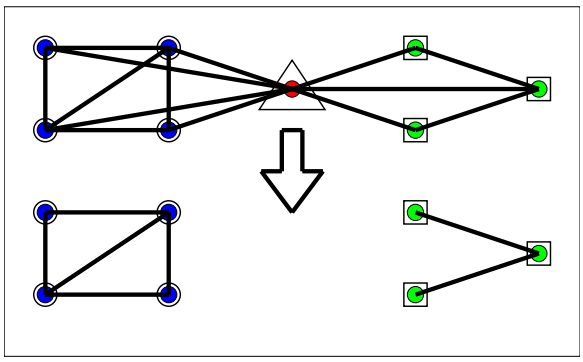


Fig. 3.1.: A toy example of clustering based name entity disambiguation

However, there are several caveats in the above simplified formulation. Particularly, the above formulation may yield numerous false alarms for various reasons even if a node u is not a multi-node. First, if the name entity u participates in multiple communities, her neighbors may form disjoint clusters. Second, the neighborhood of

the entity u may have several distinct clusters considering the temporal axis, which happens if u changes job, institution or affiliation. False negatives also occur, though with a lesser likelihood. For a multi-node u, if the significant parts of the collaboration activities of u comprise of only one name entity, the remaining weaker entities under u contribute poorly in the score $s(u)$, which may prevent from categorizing u as a multi-node. Another challenge is that due to the power-law behavior of a typical collaboration graph G, the neighborhood graph G_u has varying size and density, which affects the comparison of the score value $s(\cdot)$ of various nodes in G. We take into consideration each of these problems in our proposed solution, as explained in the following sections.

3.3 Methods

We assume that a collaboration network $G = (V, E)$ and a ego node u is given, where $u \in V$ represents an entity reference, which in a real-life scenario may be linked to multiple persons. From G and u, we construct the ego network of u, $G_u = (V_u, E_u) \subseteq G$. For each edge in G_u, say $e_u = (v, w) \in E_u$, $T(e_u)$ represents the set of collaboration events between v and w that are captured by G_u; i. e. $T(e_u) = \{\langle n_i, t_i \rangle\}_{1 \leq i \leq |T|}$, here, n_i is the number of collaboration events at time t_i. From $T(e_u)$, we compute a similarity value between v and w under G_u using an exponential decay function as shown below:

$$W_{G_u}(v, w) = \sum_{i=1}^{|T|} n_i \times \exp \frac{-(t_{\max} - t_i)}{\tau}$$

In the above equation, t_{max} denotes the most recent time when a collaboration event happened between any two vertices in G_u. τ is a tuning parameter that one can use to control the rate of the decay. On academic collaboration networks, for which the time unit is year, we use τ ranging between 5 to 10. Empirical results on such networks also show that the performance of the system is not sensitive to the choice of τ.

Example: Given $G_u = (V_u, E_u)$; $v, w \in V_u$. Let us assume $t_{max} = 2014$, v and w have 2 collaboration events in 2014, 3 collaboration events in 2013 and 4 collaboration events in 2010; thus, $T((v,w)) = \{\langle 2, 2014\rangle, \langle 3, 2013\rangle, \langle 4, 2010\rangle\}$. By setting $\tau = 5$, we get, $W_{G_u}(v,w) = 2 \times \exp\frac{-(2014-2014)}{5} + 3 \times \exp\frac{-(2014-2013)}{5} + 4 \times \exp\frac{-(2014-2010)}{5} \approx 6.25$.

3.3.1 Obtaining Cluster Quality Score

Given the collaboration network, G and a specific vertex u, the next step of our method is to construct G_u—the ego network of u. Then, we cluster the graph $G_u \setminus \{u\}$ using a graph clustering algorithm. The objective of this clustering is to group u's neighbors in different clusters such that the cross-edges between different clusters are minimized. The reason of removing u is to find whether the neighbors of u are strongly connected by themselves without using u as an intermediate vertex.

We utilize Markov Clustering (MCL) for the clustering task. MCL uses the graph's natural transition probability matrix to cluster a graph by combining random walks with two alternating operations (expansion and inflation) [99]. There are several reasons for the choice of MCL. First, MCL is one of the fastest graph clustering methods; other competing methods, such as, spectral clustering and its variants [100] compute eigenvectors of the graph Laplacian, which could be costly for large graphs. For our work, we found that for a graph with several thousands vertices, MCL finishes with good clustering results within a few seconds. Second, MCL does not require the number of clusters as one of the input parameters, which works well for our setting as we have no information regarding the number of communities in which the node u participates. Finally, MCL is robust against the choice of parameters. It has only one parameter, called *inflation*, which we set to the default value in all of our experiments.

3.3.1.1 Normalized Cut based Score

For our task, we are mainly interested in obtaining a score reflecting the quality of clustering. There are various evaluation criteria for clustering, including ratio cut,

normalized cut [99] and modularity [101]. Among these, we choose the normalized cut based clustering score as it reflects the ratio of the similarity weight-sum of the inter-cluster edges and the same for all the edges in the graph. The equation of normalized cut score for a node u is shown below:

$$NC = \sum_{i=1}^{k} \frac{W(C_i, \overline{C_i})}{W(C_i, C_i) + W(C_i, \overline{C_i})} \tag{3.1}$$

where $W(C_i, C_i)$ denotes the sum of weights of all internal edges, $W(C_i, \overline{C_i})$ is the sum of weights for all the external edges and k is the number of clusters in the graph. We compute NC by independent calculation after we obtain the clusters of G_u using the MCL algorithm.

For a node u, the normalized cut (NC) score denotes the clustering tendency of the neighbors of u. If this value is high, then the clustering tendency of u's neighbors is poor, so u is less likely to be a multi-node. On the other hand, if this value is small, then u's neighbors are well clustered and u has a high probability to be a multi-node. For example, in the bibliographic domain, if a multi-node u in a co-authorship network represents two researchers who share the same name, with a high likelihood, they will have a distinct set of co-authors. After clustering the graph $G_u \setminus \{u\}$ using MCL, we expect to obtain two dominant clusters that are disconnected (or very sparsely connected), each representing the set of co-authors of each of the two researchers. One problem with the NC-score is that it is not size invariant, i. e., if a node u has many clusters, its NC-score is large, so this score is not that useful when we compare the NC-scores of many nodes to find the one which is likely to be a multi-node. In order to address this issue, we normalize the score by the number of clusters as shown below:

$$NC\text{-}score = \frac{NC}{k} \tag{3.2}$$

where k is the total number of clusters that we obtain using MCL, given the ego network of u, G_u.

3.3.2 Obtaining Temporal Mobility Score

NC-score is a good metric to represent the degree at which a given anonymized entity is pure. However, for many real-life datasets, this score yields many false positives. Assume, a vertex in a social network represents one real-life entity, but its collaboration network evolves over time because of entity mobility. For such a vertex, the NC-score is small due to disjoint clusters of neighbors that are formed as the entity moves along with the time; this leads to a false positive prediction that the entity is likely to be a multi-node. Since the anonymized collaboration data has the time stamp of the collaboration events, we use this information to obtain a second score that we call Temporal Mobility (TM) score. This score indicates how likely is that the specific entity has moved along with the time. Note that, temporal mobility is not a new concept, it has been studied by social scientists earlier; for instance, there are works to understand the temporal mobility of academic scientists as they change their jobs [102].

To compute the likelihood that a given vertex, u, has experienced temporal mobility, we obtain the TM-score (temporal mobility score) of u. This score reflects the extent by which the node u collaborates with different neighbor clusters (obtained using MCL) at a distinct time range. Below we discuss the process that computes the TM-score of a node u.

Let's assume that $C_{i:1\leq i\leq k}$ are k different clusters that we obtain from MCL during the cluster quality computation stage. To model u's collaboration with a cluster C_i, we first obtain a vector, Z_i of size $|T|$ (the number of distinct time intervals in u's collaboration history), in which each entry denotes the total number of collaboration events between u and the members of C_i at that specific time interval. If a collaboration event with u consists of $l(\geq 2)$ actors (besides u), each of the actors in cluster C_i contributes $1/l$ to the appropriate entry in Z_i. Say, u publishes a paper with l co-authors at year t. After clustering, m of the co-authors out of the l co-authors of that paper belong to the cluster C_i, this event additively contributes to Z_i vector's

t'th position by m/l. Thus for a multi-party events, a collaboration event with u can be distributed among different clusters, in case u's accomplices for this event are distributed among different clusters. Thus, by iterating over all the collaboration events of u, we compute k different vectors, $Z_i : 1 \leq i \leq k$, each such vector corresponds to u's collaboration with the entities in one of the clusters, C_i. Finally, we use centered moving average to smooth the Z_i vectors.

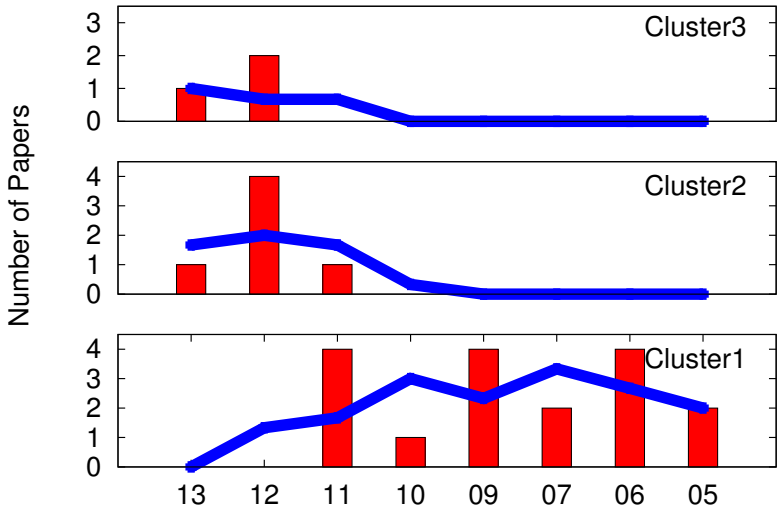


Fig. 3.2.: Temporal mobility example

For the purpose of illustration, we present an example of temporal mobility of an entity in Figure 3.2. In this Figure we can observe 3 histograms; each histogram shows the yearly count of collaboration events that this entity has with other entities in one of his neighbor-clusters. We also present the same information after smoothing with the centered moving average. We draw the histograms in this figure using real-life data of a researcher in the DBLP academic collaboration network. In the DBLP data, all the authorship records of this researcher point to a single name reference. However, when we use MCL clustering, we obtain 3 dominant clusters for this person, with almost no inter-cluster edges leading to a small NC-score. This suggests that this entity has a high chance to be a multi-node. But, as shown in Figure 3.2, the

association pattern of this entity with the three different clusters suggests temporal mobility. During 2005-2010 time period, the entity has almost dedicated association with the entities in Cluster 1; which is followed by a divided association among the entities in Cluster 2 and Cluster 3 for the time period 2011-2013.

The name of the entity that we discuss here is Dr. Honglak Lee. The first cluster corresponds to his collaboration with his co-authors when he was a PhD student in the Stanford University. The second cluster denotes his collaboration with his PhD students at the University of Michigan. The third cluster represents his collaboration with his colleagues in the same university. This example illustrates how temporal mobility of an entity can lead to a false positive multi-node when only NC-score is used for this qualification.

3.3.2.1 Temporal Mobility Score using KL Divergence

Our main objective in this step is to obtain the Temporal Mobility (TM) score of a node u after we obtain u's cluster-wise collaboration vectors, Z_i. For this we convert each of the smoothed Z_i vectors to a discrete probability distribution by normalizing these vectors appropriately. Thus, each of u's neighbor-clusters are represented by a discrete probability vector. To compute the temporal divergence of these clusters we use Kullback-Leiber (KL) divergence which is a measure of divergence between two probability distributions. $D(P \parallel Q)$ denotes the KL divergence between two probability distributions P, Q, and it is defined on a finite set χ as below:

$$D(P \parallel Q) = \sum_{x \in \chi} P(x) \log \frac{P(x)}{Q(x)}$$

This value is large, if the two distributions are different, and vice-versa. Note that, $D(P \parallel Q)$ is an asymmetric metric. For our task we use symmetric KL divergence which for the distributions P and Q are $D(P \parallel Q) + D(Q \parallel P)$. Also, to avoid the scenario when the discrete distributions of P and Q contain a zero element, we adopt Laplace correction that assigns a small probability (0.01) to those entries.

Now, TM-score of u is simply the weighted average value of the symmetric KL divergence of all pairs of Z_i (normalized to 1) vectors. For each pair, Z_i, and Z_j, we first compute $D(Z_i \parallel Z_j)$ and $D(Z_j \parallel Z_i)$. The weight of the divergence between Z_i and Z_j, is denoted as $w(Z_i, Z_j)$; we use the sum of the number of events in the cluster C_i and C_j as this weight. Using such weighting, the KL-divergence between dominant clusters contribute more in the TM-score. Similar to the case of NC-score, we also normalize TM-score by the number of clusters, so that different nodes with diverse number of clusters can be compared. The overall computation can be shown using the following equation:

$$TM\text{-}score = \frac{\sum_{i=1}^{k-1}\sum_{j=i+1}^{k} w(Z_i, Z_j) \cdot \Big(D(Z_i \parallel Z_j) + D(Z_j \parallel Z_i) \Big)}{k \times \sum_{i=1}^{k-1}\sum_{j=i+1}^{k} w(Z_i, Z_j)} \tag{3.3}$$

The higher the value of TM-score of an entity, the higher the likelihood that the node is not a multi-node. Rather it has experienced the temporal mobility phenomenon along its overall time intervals.

3.3.3 Linear Model using NC-score and TM-Score

We can use NC-score and TM-score for unsupervised learning. For an unsupervised case, we simply predict a score $s(u)$ for a node u. The higher the score, the more likely that the node is a pure node. For this we use a linear model with only one model parameter α, which is positive, because both NC-score and TM-score have larger values for a pure node and smaller value for a multi-node. Thus, the score $s(u)$ of a node u is simply:

$$s(u) = NC\text{-}score(u) + \alpha \cdot TM\text{-}score(u) \tag{3.4}$$

The model parameter α should be set manually depending on the nature of the dataset. In co-authorship networks, a small α value in the range from 0.1 to 0.2 works well. The benefit of this unsupervised method is that we can simply work on

a small collection of nodes independently. By sorting the s-score of those nodes in increasing order, we can identify the top-ranked nodes that are suspected of being a multi-node.

Algorithm 1 Unsupervised-Disambiguation(G, u)

Input: G, u
Output: $s(u)$
1: Construct the ego network G_u of u using the similarity weight between vertices
2: Remove u from G_u and apply MCL to get k clusters $\{C_i\}_{1 \leq i \leq k}$
3: Using Equation 3.1 and Equation 3.2, compute NC-score
4: For each cluster C_i, compute normalized Z_i vector using timestamp of association
5: Use Equation 3.3 to compute TM-score
6: **return** $s(u) = NC\text{-}score(u) + \alpha \cdot TM\text{-}score(u)$

3.4 Pseudo-code and Complexity Analysis

The pseudo-code of the entire process for the unsupervised setup is given in Algorithm 1. It takes an input graph G and a specific ego node u as input and generates the numeric score of u, $s(u)$ as output. Line 1 computes the similarity weights for the edges of the ego network of u, G_u. Line 2 removes u from G_u, and applies MCL clustering method to cluster the similarity graph G_u. We assume that this clustering yields k clusters, $\{C_i\}_{1 \leq i \leq k}$. From these clusters, Line 3 obtains the normalized cut based score. For each cluster (say, C_i), Line 4 computes the temporal collaboration vector Z_i of each cluster which represents the association weight between the entities in a cluster and u over the time axis. Line 5 obtains the TM-score using temporal mobility model that we discuss in Section 3.3.2. Line 6 returns the desired score for the node u. A high value of this score is more likely to represent a pure node and a small value makes the node more likely to be a multi-node. Thus our method can be used as a pre-filter to identify a small set of nodes that are more likely candidate for being a multi-node.

Given the collaboration network G and a specific vertex u as input, generation of ego network $G_u(V_u, E_u)$ takes $\mathcal{O}(|V_u| + |E_u|)$ time. The computational complexity of

MCL algorithm is $\mathcal{O}(t \cdot |V_u|^3)$ in the worst case, where t is the number of iterations until convergence. The steps in Line 3 and Line 4 read the similarity matrix of G_u using an adjacency list representation; thus the complexity of these steps is roughly $\mathcal{O}(|V_u| + |E_u|)$. The KL divergence computation in Line 5 uses Equation 3.3 to compute $TM\text{-}score$, which has a cost of $\mathcal{O}(cK^2)$, where K is the number of clusters after MCL clustering. Thus, the overall time complexity of Algorithm 1 is $\mathcal{O}(|V_u|^3)$. However, note that the above complexity bound is over the size of the ego network instead of the entire collaboration network G, which makes the proposed method very efficient. To verify the efficiency of our method, we also present the running time of our method in Section 3.5.6 over a set of real-life networks.

3.4.1 Supervised Classification Setup

In a supervised classification setting, we can use the NC-score and the TM-score as classification features. For this, we first build a training dataset in which the exact labels (positive for a multi-node, negative otherwise) of each of the instances are known. Then we can use any of our favorite classification methods to build a model, which can later be used for predicting the label for an unknown data instance. Supervised classification setup enables adding of more features for the classification task. So, in this setup, we also consider centrality-based graph topology features, in addition to the NC-score and TM-score.

We consider four distinct centrality-based features [99]: degree centrality, betweenness centrality, closeness centrality, and eigenvector centrality. For a given node u, one can always compute the centrality values of u considering the entire collaboration network, however such a method is not scalable, so we compute u's centrality within its ego-network G_u. In G_u, u is the central node by construction, but more so if u is a multi-node. This is because when u is a multi-node, it naturally has a higher than usual degree in G_u leading to a high degree centrality value for u. Also, when u is a multi node, G_u is composed of many disjoint clusters and shortest paths among the

nodes in distinct clusters must go through u, leading to a high betweenness centrality for u. Similar arguments can also be made for other centrality metrics. However, one potential issue of computing centrality within G_u is that for different nodes, their centrality values in their respective ego networks are not comparable, so we normalize u's centrality score over the centrality score of all nodes in G_u as below:

$$Centrality\text{-}Score(u) = \frac{C(u)}{\sum_{v \in G_u} C(v)}; \tag{3.5}$$

where, $C(x)$ is the centrality value of a node x in G_u. We will show in experimental results that considering centrality metrics as feature improves the prediction performance.

3.5 Experiments and Results

A key challenge of working on the name entity disambiguation task is to find a real-life labeled dataset for evaluating the performance of the proposed solution. There exist real-life collaboration datasets, such as email or phone networks, but they are anonymized for security concern, so we can't really obtain the true ambiguity label of the nodes in such networks. Hence, these datasets are not useful for evaluating the entity disambiguation task. For our experiments, we use two well known bibliographic datasets, DBLP and Arnetminer. Both of these datasets are leading repositories for bibliographic information of computer science conferences and journals. Also, the ambiguity label of a scientist in any of these networks can be determined by manual inspection of the papers published by that scientist.

From both datasets, we select 150 researchers such that half of them are pure nodes (negative cases), and the rest of them are multi-nodes (positive cases). We try to make the datasets as representative as possible by choosing a mix of senior and early career researchers. To assign the label for a selected researcher, we manually examine her bibliographic records and also her webpage profile. Besides, in DBLP, name disambiguation ground truth is already available for a few of the high profile

researchers. We use those ground truths to double-check our manual labeling. The final dataset is anonymized by mapping each researcher to a unique id.

The objective of our experiments with these datasets is to verify whether our method can distinguish the set of multi-nodes from the pure nodes. For this validation we use both supervised and unsupervised methodologies. We also compare our method with the existing state-of-the-art to show that our method is superior than that both in terms of speed and accuracy. Besides these, we also perform experiments to analyze the sensitivity of our method as the parameters vary.

Our method has only a few parameters, many of which we keep fixed. The first, τ, denotes the exponential decay rate, which is used while computing the similarity between a pair of entities in the input network before the clustering step. We fix the τ value to 5 for all of our experiments since the performance remains stable as we vary τ for both the datasets. For the clustering of the collaboration network, we use the MCL clustering method. We use the code provided by the inventor of MCL and set the inflation value of MCL to 1.4 (as is recommended by the inventor) for all of our experiments. For the other data processing, we write our own code using Python. The experiments are performed on a 2.1 GHz laptop with 4GB memory running Linux operating system.

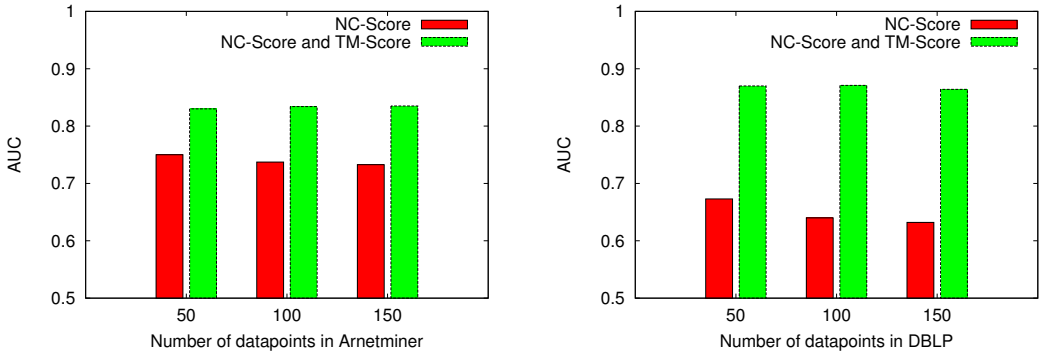


Fig. 3.3.: Unsupervised disambiguation experimental results: (a) on Arnetminer and (b) on DBLP

3.5.1 Evaluation of Unsupervised Disambiguation

In unsupervised disambiguation, we do not train a model using a training dataset, rather we use a linear function (Equation 3.4) to obtain the s-score. For evaluating the performance of unsupervised disambiguation, in this experiment we compute the s-score of each of the 150 researchers in the DBLP and Arnetminer datasets using Equation 3.4. We use an α value of 0.1 for Arnetminer and 0.2 for the DBLP dataset. The choice of α is fixed by comparing the performance of our method on a small validation dataset by varying α between 0 and 1. We use AUC (the area under the ROC curve) as the evaluation metric of this experiment which we obtain as below.

We sort the s-score of the 150 researchers in an increasing order and use each of the s-scores (in that order) as the threshold of our prediction to obtain a sequence of TPR (true positive rate) and FPR (false positive rate) pairs. From these (TPR, FPR) datapoints, we draw the ROC curve and subsequently compute the AUC value of our prediction. In this case, the AUC value is essentially the probability that the s-score of a random multi-node (positive data instance) is smaller than the s-score of a random negative data instance. The baseline value for the AUC is 0.5 and the best value of AUC is 1. The former case happens if the s-scores of positive and negative instances are non-distinguishable; on the other hand, the latter case happens when all the s-scores of the positive instances are smaller than all the s-scores of the negative instances. For the DBLP dataset and the Arnetminer dataset, the AUC value that our method achieves is 0.86 and 0.83 respectively, which, for AUC, are generally considered as excellent.

To show the variance of performance for inputs of different sizes, we construct 2 additional datasets, which are uniformly chosen random subset of the original dataset. These two datasets have 50 and 100 data instances respectively. For these datasets, we compute the AUC value as we described above. We repeat the above random dataset creation process for ten times and compute the average AUC value for both datasets. We show the AUC comparison among these datasets in Figure 3.3(a) and 3.3(b) for

the cases of Arnetminer and DBLP, respectively. As we can see for the Arnetminer case, the AUC value is almost constant (0.83) for all the three datasets with varying sizes. For DBLP, the datasets with 50 and 100 instances achieve an AUC value of 0.87, whereas the entire dataset with 150 instances achieves an AUC value of 0.86.

Table 3.1.: Comparison between our method and [37] using classification accuracy (%) on 10-fold cross-validation

Method	Kernel	DBLP	Arnetminer
Our method using NC and TM features	Linear	72.50	65.60
	Radial basis	72.31	68.82
	Sigmoid	71.90	66.20
Our method using NC, TM and Graph Centrality features	Linear	75.60	67.00
	Radial basis	74.71	69.42
	Sigmoid	75.30	70.03
Method proposed in [37]	GL-3	40.67	43.62
	GL-4	41.33	44.98
	SP	48.22	47.67

Table 3.2.: Comparison between our method and [37] using AUC on 10-fold cross-validation

Method	Kernel type	DBLP	Arnetminer
Our method using NC and TM features	Linear	0.80	0.76
	Radial basis	0.79	0.75
	Sigmoid	0.79	0.75
Our method using NC, TM and Graph Centrality features	Linear	0.83	0.80
	Radial basis	0.82	0.79
	Sigmoid	0.80	0.78
Method proposed in [37]	SP	0.62	0.61
	GL-3	0.63	0.62
	GL-4	0.64	0.62

In Figure 3.3(a) and Figure 3.3(b) we also show experimental results that highlight the contribution of TM-score in our model. For this we compare the AUC value that we obtain using TM-score and without using TM-score; the second case can be obtained by setting α=0 in Equation 3.4. The AUC value of these two cases for

datasets of different sizes are shown using green (dotted box) and red (solid box) bar plots, respectively. As we can see, for both datasets TM-score significantly improves the AUC score for the cases of all different sizes. For DBLP, the improvement is particularly significant; for the entire dataset (150 instances), the AUC without and with TM-score is 0.63 and 0.86 respectively. We guess that the reason for such dramatic improvement using TM-score is due to the fact that we use academic collaboration datasets, in such a domain temporal mobility occurs rather frequently.

Table 3.3.: Precision of multi-node class @ top-k(%) for DBLP dataset

Method	Kernel Type	Prec @10%	Prec @15%	Prec @20%
Using NC and TM features	Linear	100	100	90
	Radial basis	100	100	90
Using NC, TM and Graph Centrality features	Linear	100	100	90
	Radial basis	100	100	90
Method proposed in [37]	SP	46.7	51.7	46.7
	GL3	33.3	41.7	36.7
	GL4	46.7	41.7	44.4

Table 3.4.: Precision of multi-node class @ top-k(%) for Arnetminer dataset

Method	Kernel Type	Prec @10%	Prec @15%	Prec @20%
Using NC and TM features	Linear	100	100	90
	Radial basis	100	100	90
Using NC, TM and Graph Centrality features	Linear	100	100	90
	Radial basis	100	100	90
Method proposed in [37]	SP	66.7	54.1	60.0
	GL3	60.0	58.3	56.7
	GL4	46.7	47.5	46.7

3.5.2 Evaluation of Supervised Disambiguation

The main objective of our work is to find the s-value of a set of nodes in an unsupervised learning setup. These values can be used for the purpose of pre-filtering a

small set of suspicious nodes which can be examined more thoroughly in a subsequent stage. However, we can also use our method in a supervised learning setup to predict whether an entity is a multi-node or not. For this, we use NC-score, TM-score, and network centrality based metrics as classification features and use SVM classification tool for classification. We use the LIBSVM library with default parameter setting. During the training phase, we use the -b option of this library to predict the probability instead of predicting the class label. This makes it easier to report the performance using AUC metric. While reporting accuracy, we simply predict the instances with a probability value higher than 0.50 as positive case (a multi-node), and the remaining as a negative case.

In Table 3.1 and 3.2, we show the accuracy and AUC value for both the datasets using a 10-fold cross validation for various kernels. As we can see, for both DBLP and Arnetminer, using these features, the best classification accuracy is achieved for the linear kernel and sigmoid kernel, which are 75.60% and 70.03%, respectively. For the case of AUC, all the kernels have almost similar performance, with the best value of 0.83, and 0.80 for DBLP and Arnetminer, respectively. Considering the fact that the method works on an anonymized network, and only use topological features, accuracy value around 75% or AUC value around 0.80 are indeed commendable.

For this setup, we also report the precision@top-k for k values equal to 10%, 15%, and 20% of the size of the test datasets. We use 3-fold cross validation for this experiment. To compute precision, we simply sort the probability output of SVM in descending order and find the precision of the model in its desired range. The results are shown in Table 3.3 and Table 3.4 for the two datasets. We see that on DBLP dataset, all the top 15% of the probability values are more than 50%, thus they are predicted as positive (multi-node) class and in real-life all those instances also belong to the true positive (multi-node) class, which yields a precision of 100%. For the case of top 20%, this value drops to 90%. The result on the Arnetminer dataset is also similar (see Table 3.4). This result shows that our method is able to place most of the true multi-nodes at the top part of its ranking table, as is desired.

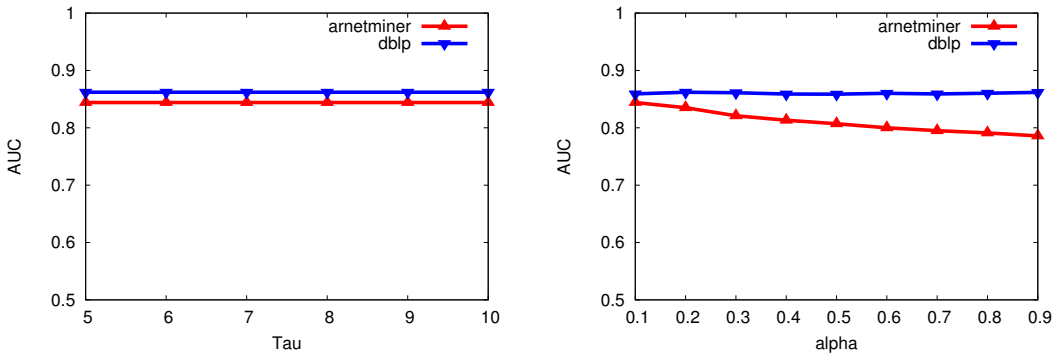
For the supervised setting, we also report the results only based on NC and TM features in terms of accuracy, AUC and precision@top-k. We can observe that adding centrality-based features improves the results in terms of accuracy and AUC. As we can see, for both DBLP and Arnetminer, using only these two features, the best classification accuracy is achieved for the linear kernel and radial basis kernel, which are 72.50% and 68.82%, respectively. For the case of AUC, all the kernels have almost similar performance, with the best value of 0.80, and 0.76 for DBLP and Arnetminer, respectively. For precision@top-k setup, the results of using NC and TM as classification features are almost the same compared with the results of adding centrality based graph topological features. Overall, the marginal improvement of using centrality based features are not that significant, which confirms that the NC-score, and the TM-score that we build are strong features for this classification task.

3.5.3 Comparison with Existing Works

The work by Hermansson et al. [37] is closely related to our work as they design a collection of graph kernels to classify multi-nodes in a supervised learning setup. Their kernels use only the graph topology, such as, graphlet counts and shortest paths, so they can be used in an anonymized network for entity disambiguation. To compare with their method, we run LIBSVM on our dataset using their best performing kernels, namely, size-3 graphlets (GL3), size-4 graphlets (GL4) and shortest path (SP) kernels. The kernel values are obtained by source code supplied by the authors. In Table 3.1, Table 3.2, Table 3.3 and Table 3.4, we compare the performance of our method that uses NC-score, TM-score, and centrality based graph topology as features with their method that uses topology based kernels, on all three performance metrics, accuracy, AUC, and precision@top-k. As we can see our method performs much better than their method on these datasets. For instance, their best kernel achieves only 48.22% accuracy on DBLP and 47.67% accuracy on Arnetminer, whereas our method achieves 75.60% and 70.03% accuracy on DBLP, and Arnetminer. Cross-validation

t-test shows that our method is significantly better (p-value 0.0051 for DBLP, and 0.0013 for Arnetminer). On AUC measure, our method obtains 0.83 an 0.80 on these datasets, whereas their method achieves a value of 0.64, and 0.62, which are much lower.

Besides improved performance, another advantage of our method over the methods in [37] is the superior running time. For a given node in the graph, the running time to compute the value of our features are only a few seconds, whereas computing graphlet kernel values is costly. In our experiments, for some of the nodes, the Matlab code provided by the authors of [37] took more than 2 days in a commodity PC.



(a) Parameter sensitivity of τ in Arnetminer and DBLP

(b) Parameter sensitivity of α in Arnetminer and DBLP

Fig. 3.4.: Parameter sensitivity of our method on two datasets

3.5.4 Study of Parameter Sensitivity

In case of unsupervised disambiguation task, our method uses two parameters that we set manually, one is the exponential decay rate (τ) for similarity computation, and the other is α value in Equation 3.4. In this experiment, we see how the performance of the model changes as we vary the value of these parameters. The result of this experiment is shown in Figure 3.4(a) and Figure 3.4(b), where we plot the AUC value for a range of parameter values. From Figure 3.4(a) we see that the performance is very stable as we vary τ. However, the performance degrades for the choice of α but not that significantly.

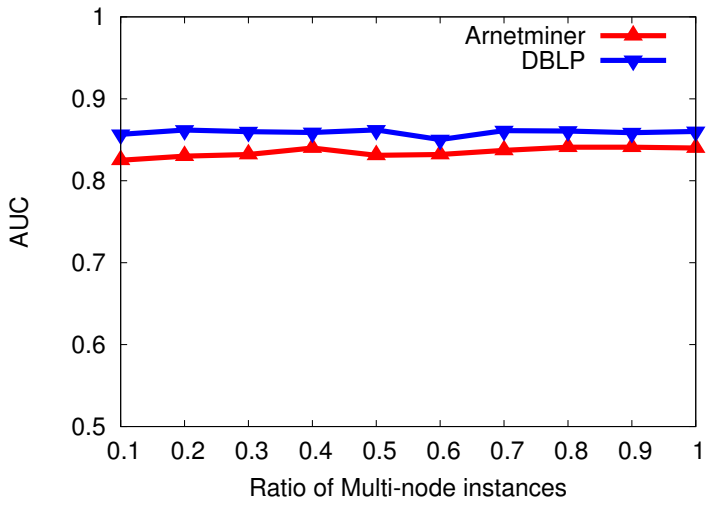


Fig. 3.5.: Bias effect in DBLP and Arnetminer

3.5.5 Study of Dataset Bias

Both our datasets are balanced, having equal number of positive and negative cases. However in real life scenario it would not be the characteristic of a wild dataset where the fraction of ambiguous entities is much lower. In this experiment we change the ratio of these two cases, to find the effect of dataset bias on the result quality. For this purpose, we change the size of positive instances and always keep the negative instances constant which is 75 negative instances during the experiments. We randomly select part of positive instances and run the method ten times and get the mean AUC as final AUC value. The result of this experiment is shown in Figure 3.5; as we can see the performance varies for different ratios but not that significantly.

3.5.6 Study of Running Time

A very desirable feature of our method is its running time. We have only two features and both of them can be calculated in a very short time. To compute the

Table 3.5.: Running time result in DBLP

Name	Number of direct neighbors	Running time (seconds)
Wei Wang	1375	2.85
Jiawei Han	523	0.55
Philip S. Yu	488	0.45
Wen Gao	531	0.50
Tao Li	603	0.58

running time, we run our unsupervised disambiguation method on 5 entities from the DBLP datasets that have the largest number of neighbors. The running time on these vertices is shown in Table 3.5. The Arnetminer dataset is smaller than the DBLP datasets, so running time on the nodes of this dataset is even smaller.

3.5.7 Real-life Case Study

Table 3.6.: Real-life case study showing prominent researchers in DBLP and Arnetminer datasets. The bold values correspond to the cases for which the prediction of our method is wrong

Name	Ground Truth	DBLP probability	Arnetminer probability
Huan Liu	+	0.80	0.68
Tao Li	+	0.86	0.75
Wei Wang	+	0.87	0.77
Tao Xie	+	0.83	0.71
Bin Li	+	0.86	0.75
Robert Allen	**+**	**0.37**	**0.23**
Tim Weninger	-	3.2e-07	0.02
Jianlin Cheng	-	5.8e-11	0.00072
Hector Gonzalez	-	7.4e-06	0.02
Xifeng Yan	-	0.38	0.42
Philip S. Yu	**-**	**0.80**	**0.70**

In Table 3.6, we show the performance of our method on some of the well-known researchers from data mining and information retrieval communities. For each of the

researchers, we denote the ground truth in the second column of the table. A positive sign stands for the fact that in DBLP and Arnetminer datasets the publication records under their names correspond to more than one real-life entity, and vice-versa. In the same table we also show the probability value that we obtain by our supervised disambiguation experiment that we discussed in Section 3.5.2. As we can see for many well known cases of multi-nodes in DBLP, such as Wei Wang, Huan Liu and Tao Li, our method correctly predicts their labels. A significant mistake (the mistaken cases are shown in bold fonts) that it makes is that it also predicts Professor Philip S. Yu to be a multi-node. This is a case of false positive, which our method is more susceptible. The reason for it is that many researchers such as, Professor Yu, have multiple disjoint communities that they maintain concurrently, so for such a researcher the NC-score is relatively small; also since his clusters do not exhibit temporal mobility, the TM-score for his case is also small. So, our method tends to predict such a person as positive. On the other hand false negative occurs in our method due to the fact that the TM-score undesirably improves the overall score of a true positive case, even though the NC-score of that case is very small. One such example is Robert Allen as we show in this table.

3.6 Chapter Summary

In this chapter, we propose a novel solution to the name disambiguation task in an anonymized network. We discuss the motivation of this task and show that our solution is useful for solving the name disambiguation task in a constrained setting, where biographical features of the actors are not available. We also discuss how our solution can be used to find a small set of suspects for whom more detailed analysis can be made in a follow-through process. Another key strength of our method is that it is robust and it uses a simple model having only two features, normalized-cut score and temporal mobility score. Nevertheless, experiments on academic collaboration networks show that our method has excellent performance. Interestingly, for these

datasets, temporal mobility score improves the prediction performance significantly. We believe that the dramatic improvement using temporal mobility feature on these datasets is due to the fact that in academic domain temporal mobility occurs rather frequently. However, due to the unavailability of ground truth datasets, we could not study whether this phenomenon presents is other networks, such as Phone call or online social networks, like Facebook. So we do acknowledge that the validity of our current work is particularly linked to academic collaboration networks and we leave the generalization of this work to networks from other domains as a future research direction.

4. NAME DISAMBIGUATION IN ANONYMIZED GRAPHS USING NETWORK EMBEDDING

4.1 Introduction

As mentioned before, the method proposed in chapter 3 can be used as a fast indicator of which entities (person nodes in the collaboration graph) to inspect more closely for ambiguity. However, the proposed approach only considers a binary prediction task, predicting whether a given person-node in the graph is ambiguous or non-ambiguous. This is far from a traditional name disambiguation task which partitions the records pertaining to a given name reference into different groups, each belonging to a unique person. Another limitation of the previous work is that it only utilizes the person-person collaboration network. However, there are other information, such as person-document association information and document-document similarity information, which can also be exploited for obtaining improved name disambiguation, yet preserving the user's privacy.

In this chapter, we solve the actual name disambiguation task by using only relational information. For a given name reference, our proposed method pre-processes the input data as three graphs: a person-person graph representing collaboration between a pair of persons, a person-document graph representing association of a person with a document and a document-document similarity graph. These graphs are appropriately anonymized, as such, the vertices of these graphs are represented by a unique pseudo-random identifier. Nodal features (such as biographical information of a person-node, or keywords of a document-node) of any of the above three graphs are not used, which makes the proposed method privacy-preserving.

In the graph representation, the name disambiguation task becomes a graph clustering task of the document-document graph, with the objective that each cluster

contains documents pertained to a unique real-life person. The traditional clustering method on a homogeneous network cannot facilitate information exchange among the three graphs, so we propose a novel representation learning model, which embeds the vertices of these graphs into a shared low dimensional latent space where name disambiguation can be solved by a hierarchical agglomerative clustering algorithm. The objective function of our representation learning task utilizes pairwise similarity ranking which is different from the typical objective functions used in the existing document embedding methods, such as LINE [51] and PTE [50]; the latter ones are based on K-L divergence between empirical similarity distribution and embedding similarity distribution. K-L divergence works over the entire distribution vector and it works well for document labeling or topic modeling, but not so for clustering. On the other hand, our objective function is better suited for a downstream clustering task because it directly optimizes the pairwise distance between similar and dissimilar documents, thus making the document vectors disambiguation-aware in the embedded space, as such, a traditional hierarchical clustering of the vectors in the embedded space generates excellent name disambiguation performance. Experimental comparison with several state-of-the-art name disambiguation methods—both traditional and network embedding-based—show that the proposed method is significantly better than the existing methods on multiple real-life name disambiguation datasets.

The key contributions of this chapter are summarized as below:

1. We propose a network embedding based solution that leverages linked structures of a variety of anonymized networks in order to represent each document into a low-dimensional vector space for solving the name disambiguation task. To the best of our knowledge, our work is the first one to adopt a representation learning framework for name disambiguation in anonymized graphs.

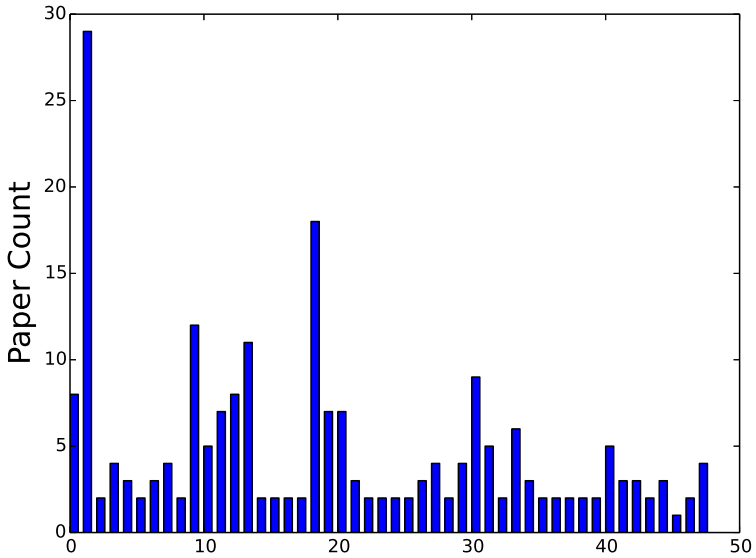


Fig. 4.1.: Paper count distribution of name reference "Lei Wang"

2. For representation learning, we present a novel pairwise ranking based objective, which is particularly suitable for solving the name disambiguation task by clustering.

3. We use two real-life bibliographic datasets for evaluating the disambiguation performance of our solution. The results demonstrate the superiority of our proposed method over the state-of-the-art methodologies for name disambiguation in a privacy-preserving setup.

4.2 Problem Formulation

We first introduce notations used in this paper. Throughout the paper, a bold uppercase letter (e.g., $\mathbf{X}$) denotes a matrix, a bold lowercase letter such as $\mathbf{x}_i$ denotes a column vector, and $(\cdot)^T$ denotes vector transpose. $\|\mathbf{X}\|_F$ is the Frobenius norm of matrix $\mathbf{X}$. A calligraphic uppercase letter (e.g., $\mathcal{X}$) is used to denote a set and $|\mathcal{X}|$ is the cardinality of the set $\mathcal{X}$.

For a given name reference a, we denote $\mathcal{D}^a = \{d_1^a, d_2^a, ..., d_N^a\}$ to be a set of N documents with which a is associated and $\mathcal{A}^a = \{a_1, a_2, ..., a_M\}$ is the collaborator set of a in $\mathcal{D}^a$, where $a \notin \mathcal{A}^a$. If there is no ambiguity we remove the superscript a in the notations of both $\mathcal{D}^a$ and $\mathcal{A}^a$ and refer the terms as $\mathcal{D}$ and $\mathcal{A}$, respectively. For illustration, in the bibliographic field, $\mathcal{D}$ can be the set of scholarly publications where a is one of the authors and A is the set of a's coauthors. In real-life, the given name reference a can be associated with multiple persons (say L) all sharing the same name. The task of name disambiguation is to partition $\mathcal{D}$ into L disjoint sets such that each partition contains documents of a unique person entity with name reference a.

Though it may appear as a simple clustering problem, name disambiguation is challenging on real-life data. This is due to the fact that it requires solving a highly class-imbalanced clustering task, as the number of documents associated with a distinct person follows a power-law distribution. We demonstrate it through an example from the bibliographic domain. In Figure 4.1, we show a histogram of paper counts of various real-life persons named "Lei Wang" in Arnetminer.As we can observe, there are a few real-life authors (dominant entities) with the name "Lei Wang" to whom the majority of the publications belong. Only a few publications belong to each of the remaining real-life authors with name "Lei Wang". Due to this severe class imbalance issue, the majority of traditional clustering methods perform poorly on this task. Sophisticated machine learning models, like the one we propose below are needed for solving this task. This example is from the bibliographic domain, but power-law distribution of possession is common in every aspect of real-life, so we expect this challenge to hold in other domains as well.

In this study, we investigate the name disambiguation problem in a restricted setup, where bibliographical features and information from external sources are not considered so that the risk of privacy violation can be reduced. Instead, we formulate the problem using graphs in which each node has been assigned an anonymized identifier, and network topological structure is the only information available. Specif-

ically, our solution encodes the local neighborhood structures accumulated from three different networks into a proposed network embedding model, which generates a k-dimensional vector representation for each document. The networks are person-person network, person-document network, and linked document network, which we formally define as below:

Definition 4.2.1 (Person-Person Network) For a given name reference x, the person-person network, denoted as $G_{pp} = (\mathcal{A}^x, E_{pp})$, captures collaboration between a pair of persons within the collection of documents associated with x. $\mathcal{A}^x$ is the collaborator set, and $e_{ij} \in E_{pp}$ represents the edge between the persons, a_i and a_j, who collaborated in at least one document. The weight w_{ij} of the edge e_{ij} is defined as the number of distinct documents in which a_i and a_j have collaborated.

The person-person network is important because the inter-person acquaintances represented by collaboration relation can be used to discriminate the set of documents of multiple real-life persons. However, the collaboration network does not account for the fact that the documents associated with the same real-life person are inherently similar; person-document network and document-document network cover for this shortcoming.

Definition 4.2.2 (Person-Document Network) Person-Document Network, represented as $G_{pd} = (\mathcal{A} \cup \mathcal{D}, E_{pd})$, is a bipartite network where $\mathcal{D}$ is the set of documents with which the name reference a is associated and $\mathcal{A}$ is the set of collaborators of a over all the documents in $\mathcal{D}$. E_{pd} is the set of edges between persons and documents. The edge weight w_{ij} between a person node a_i and document d_j is simply defined as the number of times a_i appears in document d_j. For a bibliographic dataset, a_i is simply an author of the document d_j and the weight $w_{ij} = 1$.

Definition 4.2.3 (Linked Document Network) Document-Document Network, represented as $G_{dd} = (\mathcal{D}, E_{dd})$, where each vertex $d_i \in \mathcal{D}$ is a document. If two documents d_i and d_j are similar (more discussion is forthcoming), we build an edge between them represented as $e_{ij} \in E_{dd}$.

There are several ways document-document similarity can be captured. For instance, one can find word co-occurrence between different documents to compute this similarity. However, we refrained from using word co-occurrence due to the privacy concern as sometimes a list of a set of unique words can reveal the identity of a person [1,20]. Instead we define document-document similarity through a combination of person-person and person-document relationships. Two documents are similar if the intersection of their collaborator-sets is large (by using person-document relationship) or if the intersection of one-hop neighbors of their collaborator-sets is large (by using both person-document and person-person relationships).

The above definition of document similarity captures two important patterns which facilitate effective name disambiguation by document clustering. First, there is a high chance for two documents to be authored by the same real-life person, if they have a large number of overlapping collaborators. Second, even if they do not have any overlapping collaborators, large overlap in the neighbors of their collaborators signals that the documents are most likely authored by the same person. For both cases, these two documents should be placed in close proximity in the embedded space. Mathematically, we denote $\mathcal{A}^1_{d_i}$ as the collaborator set of d_i. Furthermore, $\mathcal{A}^2_{d_i}$ is the set of collaborators by extending $\mathcal{A}^1_{d_i}$ with all neighbors of the persons in $\mathcal{A}^1_{d_i}$, namely $\mathcal{A}^2_{d_i} = \mathcal{A}^1_{d_i} \cup \{\mathcal{NB}_{G_{pp}}(b)\}_{b \in \mathcal{A}^1_{d_i}}$, where $\mathcal{NB}_{G_{pp}}(b)$ is the set of neighbors of node b in person-person network G_{pp}. Then the document similarity between d_i and d_j in the graph G_{dd} is simply defined as $w_{ij} = |\mathcal{A}^2_{d_i} \cap \mathcal{A}^2_{d_j}|$.

Based on our problem formulation, the name disambiguation solution consists of two phases: (1) document representation (2) disambiguation. We discuss them below:

Given a name reference a, its associated document set $\mathcal{D}^a$ (which we want to disambiguate) and the collaborator set $\mathcal{A}^a$, the document representation phase first constructs corresponding person-person network G_{pp}, person-document bipartite network G_{pd}, and linked document network G_{dd}. Then our proposed document representation model combines structural information from these three networks to generate a k-dimensional document embedding matrix $\mathbf{D} = [\mathbf{d}_1^T, ..., \mathbf{d}_N^T] \in \mathbb{R}^{N \times k}$.

The disambiguation phase takes the document embedding matrix $\mathbf{D}$ as input and applies the hierarchical agglomerative clustering (HAC) with group average merging criteria to partition N documents in $\mathcal{D}^a$ into L disjoint sets with the expectation that each set is composed of documents of a unique person entity sharing the name reference a. At this stage, L is a user-defined parameter which we match with the ground truth during the evaluation phase. In real-life, a user needs to tune the parameter L which can easily be done with HAC, because HAC provides hierarchical organization of clusters at all levels starting from a single cluster upto the case of single-instance cluster, and a user can recover clustering for any value of L as needed without additional cost. Also, across different L values the cluster assignment of HAC is consistent (i.e., two instances that are in the same cluster for some L value will remain in the same cluster for any smaller L value), which further helps in choosing an appropriate L value.

4.3 Method

In this section, we discuss our proposed representation learning model for name disambiguation. Our goal is to encode the local neighborhood structures captured by the three networks (see Definitions 4.2.1 4.2.2 4.2.3) into the k-dimensional document embedding matrix with strong name disambiguation ability.

4.3.1 Model Formulation

The main intuition of our network embedding model is that neighboring nodes in a graph should have more similar vector representation in the embedding space than non-neighboring nodes. For instance, in a linked document network, the affinity between two neighboring vertices d_i and d_j, i.e., $e_{ij} \in G_{dd}$ should be larger than the affinity between two non-neighboring vertices d_i and d_t, i.e., $e_{it} \notin G_{dd}$. The affinity score between two nodes d_i and d_j in G_{dd} can be calculated as the inner product of their corresponding embedding representations, denoted as $S^{dd}_{ij} = \mathbf{d}_i^T \mathbf{d}_j$. More

specifically, we model the probability of preserving ranking order $S^{dd}_{ij} > S^{dd}_{it}$ using the logistic function $\sigma(x) = \frac{1}{1+e^{-x}}$. Mathematically,

$$P\big(S^{dd}_{ij} > S^{dd}_{it} | \mathbf{d}_i, \mathbf{d}_j, \mathbf{d}_t\big) = \sigma\big(S^{dd}_{ijt}\big) \tag{4.1}$$

where S^{dd}_{ijt} is defined as below:

$$\begin{aligned} S^{dd}_{ijt} &= S^{dd}_{ij} - S^{dd}_{it} \\ &= \mathbf{d}_i^T \mathbf{d}_j - \mathbf{d}_i^T \mathbf{d}_t \end{aligned} \tag{4.2}$$

As we observe from Equation 4.1, the larger S^{dd}_{ijt}, the more likely ranking order $S^{dd}_{ij} > S^{dd}_{it}$ is preserved. By assuming all the ranking orders generated from the linked document network G_{dd} to be independent, the probability $P(> |\mathbf{D})$ of all the ranking orders being preserved given the document embedding matrix $\mathbf{D} \in \mathbb{R}^{N \times k}$ is defined as below:

$$\begin{aligned} P(> |\mathbf{D}) &= \prod_{\substack{(d_i,d_j) \in \mathcal{P}_{G_{dd}} \\ (d_i,d_t) \in \mathcal{N}_{G_{dd}}}} P\big(S^{dd}_{ij} > S^{dd}_{it} | \mathbf{d}_i, \mathbf{d}_j, \mathbf{d}_t\big) \\ &= \prod_{\substack{(d_i,d_j) \in \mathcal{P}_{G_{dd}} \\ (d_i,d_t) \in \mathcal{N}_{G_{dd}}}} \sigma\big(S^{dd}_{ijt}\big) \\ &= \prod_{\substack{(d_i,d_j) \in \mathcal{P}_{G_{dd}} \\ (d_i,d_t) \in \mathcal{N}_{G_{dd}}}} \sigma\big(S^{dd}_{ij} - S^{dd}_{it}\big) \end{aligned} \tag{4.3}$$

where $\mathcal{P}_{G_{dd}}$ and $\mathcal{N}_{G_{dd}}$ are positive and negative training sets in linked document network, represented as G_{dd}.

From the Equation 4.3, the goal is to seek the document latent representation $\mathbf{D}$ for all nodes in linked document network G_{dd}, which maximizes $P(>|\mathbf{D})$. For the computational convenience, we minimize the following sum of negative log-likelihood objective, which is shown as follows:

$$\begin{aligned} OBJ_{dd} &= \min_{\mathbf{D}} -\ln P(>|\mathbf{D}) \\ &= -\sum_{\substack{(d_i,d_j)\in\mathcal{P}_{G_{dd}} \\ (d_i,d_t)\in\mathcal{N}_{G_{dd}}}} \ln P\big(S_{ij}^{dd} > S_{it}^{dd}|\mathbf{d}_i,\mathbf{d}_j,\mathbf{d}_t\big) \\ &= -\sum_{\substack{(d_i,d_j)\in\mathcal{P}_{G_{dd}} \\ (d_i,d_t)\in\mathcal{N}_{G_{dd}}}} \ln \sigma(S_{ijt}^{dd}) \\ &= -\sum_{\substack{(d_i,d_j)\in\mathcal{P}_{G_{dd}} \\ (d_i,d_t)\in\mathcal{N}_{G_{dd}}}} \ln \sigma\big(S_{ij}^{dd} - S_{it}^{dd}\big) \end{aligned} \tag{4.4}$$

The formulation shown in Equation 4.4 constructs a probabilistic framework for distinguishing between neighbor nodes and non-neighbor nodes in a linked document network by preserving a ranking order objective function.

Using the identical argument, the objective functions for capturing person-person and person-document relations are given as below:

$$\begin{aligned} OBJ_{pp} &= \min_{\mathbf{A}} -\ln P(>|\mathbf{A}) \\ &= -\sum_{\substack{(a_i,a_j)\in\mathcal{P}_{G_{pp}} \\ (a_i,a_t)\in\mathcal{N}_{G_{pp}}}} \ln \sigma(S_{ij}^{pp} - S_{it}^{pp}) \end{aligned} \tag{4.5}$$

$$
\begin{aligned}
OBJ_{pd} &= \min_{\mathbf{A},\mathbf{D}} -\ln P(> |\mathbf{A},\mathbf{D}) \\
&= -\sum_{\substack{(d_i,a_j)\in\mathcal{P}_{G_{pd}} \\ (d_i,a_t)\in\mathcal{N}_{G_{pd}}}} \ln\sigma(S_{ij}^{pd} - S_{it}^{pd})
\end{aligned}
\tag{4.6}
$$

where $\mathbf{A} \in \mathbb{R}^{M\times k}$ can be thought as the person embedding matrix and M is the number of persons in the collaborator set $\mathcal{A}$. S_{ij}^{pp} represents the affinity score between two nodes a_i and a_j in collaboration graph G_{pp}, and S_{ij}^{pd} denotes the affinity score between two nodes d_i and a_j in heterogeneous bipartite graph G_{pd}. Finally, $\mathcal{P}_{G_{pp}}$ and $\mathcal{N}_{G_{pp}}$ are positive and negative training sets in G_{pp}, $\mathcal{P}_{G_{pd}}$ and $\mathcal{N}_{G_{pd}}$ are positive and negative training sets in G_{pd} respectively.

The goal of proposed network embedding framework is to unify these three types of relations together, where the person and document vertices are shared across these three networks. An intuitive manner is to collectively embed these three networks, which can be achieved by minimizing the following objective function:

$$
OBJ = \min_{\mathbf{A},\mathbf{D}} -OBJ_{pp} - OBJ_{pd} - OBJ_{dd} + \lambda Reg(\mathbf{A},\mathbf{D}) \tag{4.7}
$$

where $\lambda Reg(\mathbf{A},\mathbf{D})$ in Equation 4.7 is a l_2-norm regularization term to prevent the model from overfitting. Here for the computational convenience, we set $Reg(\mathbf{A},\mathbf{D})$ as $\|\mathbf{A}\|_F^2 + \|\mathbf{D}\|_F^2$. Such pairwise ranking loss objective is in the similar spirit to the Bayesian Personalized Ranking [103, 104], which aims to predict the interaction between users and items in recommender system domain.

4.3.2 Model Optimization

We use stochastic gradient descent (SGD) algorithm for optimizing Equation 4.7. Specifically, in each step we sample the training instances involved in person-person, person-document, and document-document relations accordingly. The sampling strategy of positive instances is based on edge sampling [50]. Specifically, for example, in

linked document network G_{dd}, given an arbitrary node d_i, we sample one of its neighbors d_j, i.e., $(d_i, d_j) \in \mathcal{P}_{G_{dd}}$, with the probability proportional to the edge weight for the model update. On the other hand, for sampling of negative instances, we utilize uniform sampling technique. In particular, given the sampled node d_i, we sample an arbitrary negative instance d_t uniformly, namely $(d_i, d_t) \in \mathcal{N}_{G_{dd}}$.

Therefore given a sampled triplet (d_i, d_j, d_t) with $(d_i, d_j) \in \mathcal{P}_{G_{dd}}$ and $(d_i, d_t) \in \mathcal{N}_{G_{dd}}$, using the chain rule and back-propagation, the gradient of the objective function OBJ in Equation 4.7 w.r.t. $\mathbf{d}_i$ can be computed as below:

$$\begin{aligned}
\frac{\partial OBJ}{\partial \mathbf{d}_i} &= -\frac{\partial \ln \sigma\big(S_{ij}^{dd} - S_{it}^{dd}\big)}{\partial \mathbf{d}_i} + 2\lambda \mathbf{d}_i \\
&= -\frac{\partial \ln \sigma\big(S_{ij}^{dd} - S_{it}^{dd}\big)}{\partial \sigma\big(S_{ij}^{dd} - S_{it}^{dd}\big)} \times \frac{\partial \sigma\big(S_{ij}^{dd} - S_{it}^{dd}\big)}{\partial\big(S_{ij}^{dd} - S_{it}^{dd}\big)} \\
&\quad \times \frac{\partial\big(S_{ij}^{dd} - S_{it}^{dd}\big)}{\partial \mathbf{d}_i} + 2\lambda \mathbf{d}_i \\
&= -\frac{1}{\sigma\big(S_{ij}^{dd} - S_{it}^{dd}\big)} \times \sigma\big(S_{ij}^{dd} - S_{it}^{dd}\big) \\
&\quad \Big(1 - \sigma\big(S_{ij}^{dd} - S_{it}^{dd}\big)\Big) \times (\mathbf{d}_j - \mathbf{d}_t) + 2\lambda \mathbf{d}_i \\
&= \left(\frac{-e^{-(\mathbf{d}_i^T \mathbf{d}_j - \mathbf{d}_i^T \mathbf{d}_t)}}{1 + e^{-(\mathbf{d}_i^T \mathbf{d}_j - \mathbf{d}_i^T \mathbf{d}_t)}}\right)(\mathbf{d}_j - \mathbf{d}_t) + 2\lambda \mathbf{d}_i
\end{aligned} \tag{4.8}$$

Using the similar chain rule derivation, the gradient of the objective function OBJ w.r.t. $\mathbf{d}_j$ and $\mathbf{d}_t$ can be obtained as follows:

$$\frac{\partial OBJ}{\partial \mathbf{d}_j} = \left(\frac{-e^{-(\mathbf{d}_i^T \mathbf{d}_j - \mathbf{d}_i^T \mathbf{d}_t)}}{1 + e^{-(\mathbf{d}_i^T \mathbf{d}_j - \mathbf{d}_i^T \mathbf{d}_t)}}\right) \times \mathbf{d}_i + 2\lambda \mathbf{d}_j \tag{4.9}$$

$$\frac{\partial OBJ}{\partial \mathbf{d}_t} = \left(\frac{-e^{-(\mathbf{d}_i^T \mathbf{d}_j - \mathbf{d}_i^T \mathbf{d}_t)}}{1 + e^{-(\mathbf{d}_i^T \mathbf{d}_j - \mathbf{d}_i^T \mathbf{d}_t)}}\right) \times (-\mathbf{d}_i) + 2\lambda \mathbf{d}_t \tag{4.10}$$

Then embedding vectors $\mathbf{d}_i$, $\mathbf{d}_j$, and $\mathbf{d}_t$ are updated as below:

$$\begin{aligned}\mathbf{d}_i &= \mathbf{d}_i - \alpha\frac{\partial OBJ}{\partial \mathbf{d}_i}\\ \mathbf{d}_j &= \mathbf{d}_j - \alpha\frac{\partial OBJ}{\partial \mathbf{d}_j}\\ \mathbf{d}_t &= \mathbf{d}_t - \alpha\frac{\partial OBJ}{\partial \mathbf{d}_t}\end{aligned} \tag{4.11}$$

where α is the learning rate.

Likewise, when the training instances come from person-person network, and person-document bipartite network, we update their corresponding gradients accordingly. We omit the detailed derivations here since they are very similar to the aforementioned ones.

Algorithm 2 Network Embedding based Name Disambiguation in Anonymized Graphs

Input: name reference a, dimension k, λ, α, L
Output: document embedding matrix $\mathbf{D}$ and its clustering membership set $\mathcal{C}$

1: Given name reference a, construct its associated $\mathcal{D}^a$, $\mathcal{A}^a$, G_{pp}, G_{pd}, G_{dd}
2: Given G_{pp}, G_{pd}, G_{dd}, construct training sample sets $\mathcal{P}_{G_{pp}}$, $\mathcal{N}_{G_{pp}}$, $\mathcal{P}_{G_{pd}}$, $\mathcal{N}_{G_{pd}}$, $\mathcal{P}_{G_{dd}}$, $\mathcal{N}_{G_{dd}}$ respectively based on edge sampling and uniform sampling techniques
3: Initialize $\mathbf{A}$ and $\mathbf{D}$ as k-dimensional matrices
4: **for** each training instance in training sample sets **do**
5: Update involved parameters using SGD as described in Section 4.3.2
6: **end for**
7: Given $\mathbf{D}$ and L, perform HAC to partition N documents in $\mathcal{D}^a$ into L disjoint sets for name disambiguation
8: **return** $\mathbf{D}$, $\mathcal{C} = \{c_1, c_2, ..., c_N\}$

4.3.3 Pseudo-code and Complexity Analysis

The pseudo-code of the proposed network embedding method for name disambiguation under anonymized graphs is summarized in Algorithm 4. The entire process

consists of two phases: network embedding for document representation and name disambiguation by clustering. Specifically, given a name reference a and its associated document set $\mathcal{D}^a$ we aim to disambiguate, we first prepare the training instances in Line 1-2. Line 3 initializes the person and document embedding matrices $\mathbf{A}$ and $\mathbf{D}$ by randomly sampling elements from uniform distribution $[-0.2, 0.2]$. Then we train our proposed network embedding model and update $\mathbf{A}$ and $\mathbf{D}$ using the training samples based on the SGD optimization in Line 4-6. Then given the obtained document embedding matrix $\mathbf{D}$ and L, in Line 7, we perform HAC to partition N documents in $\mathcal{D}^a$ into L disjoint sets such that each partition contains documents of a unique person entity with name reference a. Finally in Line 8, we return document embedding matrix $\mathbf{D}$ and its clustering membership set $\mathcal{C} = \{c_1, ..., c_i, ..., c_N\}$ for evaluation, where $1 \leq c_i \leq L$.

For the time complexity analysis, for the document embedding, when the training sample is $(d_i, d_j) \in \mathcal{P}_{G_{dd}}$, as observed from Equations 4.8, 4.9 and 4.11, the cost of calculating gradient of OBJ w.r.t. $\mathbf{d}_i$ and $\mathbf{d}_j$, and updating $\mathbf{d}_i$ and $\mathbf{d}_j$ are both $\mathcal{O}(k)$. Similar analysis can be applied when training instances are from $\mathcal{P}_{G_{pp}}$, $\mathcal{N}_{G_{pp}}$, $\mathcal{P}_{G_{pd}}$, $\mathcal{N}_{G_{pd}}$, $\mathcal{N}_{G_{dd}}$. Therefore, the total computational cost is $\big(2 * |\mathcal{P}_{G_{pp}}| + 2 * |\mathcal{P}_{G_{pd}}| + 2 * |\mathcal{P}_{G_{dd}}|\big)\mathcal{O}(k)$. For the name disambiguation, the computational cost of hierarchical clustering is $\mathcal{O}(N^2 logN)$ [99]. So the total computational complexity of Algorithm 2 is $\big(2 * |\mathcal{P}_{G_{pp}}| + 2 * |\mathcal{P}_{G_{pd}}| + 2 * |\mathcal{P}_{G_{dd}}|\big)\mathcal{O}(k) + \mathcal{O}(N^2 logN)$.

4.3.4 Mining Hard Negative Training Instance

In the proposed pairwise ranking based embedding framework, we need to construct negative training instances. Since the optimization algorithm we utilize is based on SGD, the sampled negative instances may affect the name disambiguation performance substantially. Therefore, motivated by existing works [105, 106], instead of uniformly sampling negative instance in each iteration, we present two extra sampling strategies to select negative training instance, which are listed as below:

1. Dynamic negative sampling [105]: this method aims to dynamically choose negative training samples from the ranked list produced by current embedding model and iteratively update the learning model. Specifically, in each iteration, we first randomly sample a set of negative instances given each of the constructed networks, namely G_{pp}, G_{pd}, G_{dd} respectively, then we apply the current embedding model on them to get the prediction scores. Finally, we select the one among the top ranked list based on their corresponding prediction scores to update the embedding vectors.

2. Adaptive sampling [106]: our goal is to use the current prediction model to define the sampling distribution and the negative instances are sampled through a pre-defined exponential distribution to update the learning model.

4.4 Experiments and Results

We perform several experiments to validate the performance of our proposed network embedding method for solving the name disambiguation task in a privacy-preserving setting using only linked data. We also compare our method with various other methods to demonstrate its superiority over those methods.

4.4.1 Datasets

A key challenge for the evaluation of name disambiguation task is the lack of availability of labeled datasets from diverse application domains. In recent years, the bibliographic repository sites, Arnetminer and CiteSeerX have published several ambiguous author name references along with respective ground truths (paper list of each real-life author), which we use for evaluation. From each of these two sources, we use 10 highly ambiguous (having a larger number of distinct authors for a given name) name references and show the performance of our method on these name references. The statistics of name references in Arnetminer and CiteSeerX datasets are shown in

Table 4.1.: Arnetminer name disambiguation dataset

Name Reference	# Documents	# Distinct Authors
Jing Zhang	160	33
Bin Yu	78	8
Rakesh Kumar	82	5
Lei Wang	222	48
Bin Li	135	14
Yang Wang	134	23
Bo Liu	93	19
Yu Zhang	156	26
David Brown	42	9
Wei Xu	111	21

Table 4.2.: CiteSeerX name disambiguation dataset

Name Reference	# Documents	# Distinct Authors
K Tanaka	174	9
M Jones	191	10
J Smith	798	26
Y Chen	848	64
J Martin	51	13
A Kumar	149	10
J Robinson	123	9
M Brown	118	13
J Lee	891	93
S Lee	1091	74

Table 4.1 and Table 4.2, respectively. In these tables, for each name reference, we show the number of documents, and the number of distinct authors associated with that name reference. It is important to understand that the name disambiguation model is built on a name reference, not on a source dataset such as, Arnetminer or CiteSeerX as a whole, so each name reference is a distinct dataset on which the evaluation is performed.

4.4.2 Competing Methods

To validate the disambiguation performance of our proposed approach, we compare it against 9 different methods. For a fair comparison, all of these methods accommodate the name disambiguation using only relational data. Among all the competing methods, Rand, AuthorList, and AuthorList-NNMF are a set of primitive baselines that we have designed. But, the remaining methods are taken from recently published works. For instance, GF, DeepWalk, LINE, Node2Vec, and PTE are existing state-of-the-art approaches for vertex embedding, which we use for name disambiguation by clustering the documents using HAC in the embedding space similar to our approach. Graphlet based graph kernel methods (GL3, GL4) are existing state-of-the-art approaches for name disambiguation in anonymized graphs. More details of each of the competing methods are given below. For each method, for a given name reference, a list of documents need to be partitioned among L (user defined) different clusters.

(1) Rand: This naive method randomly assigns one of existing classes to the associated documents.

(2) AuthorList: Given the associated documents, we first aggregate the author-list of all documents in an author-array, then define a binary feature for each author, indicating his presence or absence in the author-list of that document. Finally we use HAC with the generated author-list as features for disambiguation task.

(3) AuthorList-NNMF: We perform Non-Negative Matrix Factorization (NNMF) on the generated author-list features the same way described above. Then the latent features from NNMF are used in a HAC framework for disambiguation task.

(4) Graph Factorization (GF) [107]: We first represent co-authorship network G_{pp} and the linked document network G_{dd} as affinity matrices, and then utilize matrix factorization technique to represent each document into low-dimensional vector. Note that GF is optimized via a point-wise regression model that minimizes a square loss function, which is substantially different from our proposed ranking loss objective.

(5) DeepWalk [49]: DeepWalk is an approach recently proposed for network embedding, which is only applicable for homogeneous network with binary edges. Given G_{pp} and G_{dd}, we use uniform random walk to obtain the contextual information of its neighborhood for document embedding [1].

(6) LINE [51]: LINE aims to learn the document embedding that preserves both the first-order and second-order proximities [2]. Note that LINE can only handle the embedding of homogeneous network and the embedding formulation and optimization are quite different from the one proposed in our work.

(7) Node2Vec [52]: Similar to DeepWalk, Node2Vec designs a biased random walk procedure for document embedding. [3].

(8) PTE [50]: Predictive Text Embedding (PTE) framework aims to capture the relations of word-word, word-document, and word-label. However, such keyword and label based biographical features are not available in the anonymized setup. Instead we utilize local structural information of both G_{pp} and G_{pd} networks to learn the document embedding. However, this approach is not able to capture the linked information among documents.

(9) Graph Kernel [37]: In this work, size-3 graphlets (GL3) and size-4 graphlets (GL4) are used to build graph kernels, which measure the similarity between documents. Then the learned similarity metric is used as features in HAC for name disambiguation. As we see, both kernels only use network topological information. [4]

4.4.3 Experimental Setting and Implementation

For each of the 20 name references, we perform name disambiguation using our proposed method and each of the competing methods to demonstrate that our proposed method is superior than the competing methods. For evaluation metric, we use Macro-F1 measure [99], which is the unweighted average of F1 measure of each class.

[1]Code is available at `http://www.perozzi.net/projects/deepwalk/`
[2]Implementation Code is available at `https://github.com/tangjianpku/LINE`
[3]We use the code from `https://github.com/aditya-grover/node2vec`
[4]The kernel values are obtained by source code supplied by the original authors

Table 4.3.: Comparison of Macro-F1 values between our proposed method and other competing methods for name disambiguation task in Arnetminer dataset (embedding dimension = 20). Paired t-test is conducted on all performance comparisons and it shows that all improvements are significant at the 0.05 level.

Name Reference	**Our Method**	Rand	AuthorList	AuthorList-NNMF	GF [107]	DeepWalk [49]	LINE [51]	Node2Vec [52]	PTE [50]	GL3 [37]	GL4 [37]	Improv.
Jing Zhang	**0.734 (0.014)**	0.192	0.327	0.463	0.669	0.654	0.651	0.312	0.458	0.318	0.329	9.7%
Bin Yu	**0.804 (0.009)**	0.201	0.371	0.283	0.610	0.644	0.643	0.531	0.399	0.489	0.504	24.8%
Rakesh Kumar	**0.834 (0.012)**	0.226	0.305	0.404	0.448	0.617	0.641	0.372	0.219	0.434	0.407	30.1%
Lei Wang	**0.805 (0.021)**	0.198	0.502	0.424	0.633	0.419	0.639	0.263	0.447	0.291	0.321	26.0%
Bin Li	**0.848 (0.016)**	0.172	0.610	0.733	0.761	0.392	0.641	0.186	0.349	0.336	0.418	11.4%
Yang Wang	**0.798 (0.011)**	0.199	0.442	0.532	0.575	0.640	0.623	0.331	0.444	0.378	0.512	24.7%
Bo Liu	0.831 (0.022)	0.215	0.482	0.740	**0.850**	0.788	0.781	0.459	0.373	0.498	0.347	-2.2%
Yu Zhang	**0.820 (0.031)**	0.186	0.519	0.566	0.565	0.454	0.658	0.196	0.385	0.369	0.305	24.6%
David Brown	**1.00 (0.00)**	0.304	0.818	0.583	0.802	0.494	**1.00**	0.221	0.575	0.603	0.698	0%
Wei Xu	**0.793 (0.014)**	0.256	0.527	0.564	0.625	0.228	0.599	0.136	0.236	0.386	0.428	26.9%

The range of Macro-F1 measure is between 0 and 1, and a higher value indicates better disambiguation performance. Besides comparison with competing methodologies, we also perform experiments to show that our method is robust against the variation of user defined parameters (specifically, embedding dimension and the number of clusters) over a wide range of parameter values. Experiments are also performed to show how the embedding model performs with each of the three types of networks (person-person, person-document, and document-document) incrementally added. Finally, we show the convergence of the learning model while performing the document embedding phase.

There are a few user defined parameters in our proposed embedding model. The first among these is the embedding dimension k, which we set to be 20. For the regularization parameter in model inference (see Section 4.3.2), we perform grid search on the validation set in the following range: $\lambda = \{0.001, 0.005, 0.01, 0.1, 1, 10\}$. For the learning rate, we fix α as 0.02. During the disambiguation stage, we use the actual number of classes L of each name reference as input to perform HAC. In addition to that, we use uniform sampling to select negative instances for SGD optimization. For both data processing and model implementation, we implement our own code in Python and use NumPy, SciPy, scikit-learn, and Networkx libraries for linear algebra, machine learning, and graph operations. We run all the experiments on a 2.1 GHz Machine with 8GB memory running Linux operating system.

Table 4.4.: Comparison of Macro-F1 values between our proposed method and other competing methods for name disambiguation task in CiteSeerX dataset (embedding dimension = 20). Paired t-test is conducted on all performance comparisons and it shows that all improvements are significant at the 0.05 level.

Name Reference	**Our Method**	Rand	AuthorList	AuthorList-NNMF	GF [107]	DeepWalk [49]	LINE [51]	Node2Vec [52]	PTE [50]	GL3 [37]	GL4 [37]	Improv.
K Tanaka	**0.706 (0.018)**	0.178	0.202	0.168	0.334	0.450	0.398	0.304	0.173	0.235	0.276	56.9%
M Jones	**0.743 (0.009)**	0.184	0.189	0.261	0.529	0.696	0.688	0.513	0.348	0.216	0.398	6.8%
J Smith	**0.503 (0.007)**	0.083	0.121	0.280	0.316	0.098	0.104	0.073	0.136	0.201	0.237	59.2%
Y Chen	0.367 (0.019)	0.069	0.325	0.355	**0.439**	0.118	0.193	0.058	0.199	0.334	0.385	-16.4%
S Lee	**0.624 (0.015)**	0.057	0.214	0.248	0.345	0.194	0.109	0.044	0.256	0.215	0.268	80.9%
J Martin	**0.898 (0.021)**	0.310	0.624	0.536	0.755	0.728	0.774	0.629	0.587	0.414	0.431	16.0%
A Kumar	**0.645 (0.006)**	0.166	0.251	0.375	0.319	0.407	0.395	0.424	0.247	0.192	0.234	52.1%
J Robinson	**0.796 (0.033)**	0.200	0.348	0.438	0.393	0.513	0.603	0.608	0.345	0.271	0.316	30.9%
M Brown	**0.741 (0.028)**	0.171	0.306	0.573	0.478	0.481	0.633	0.211	0.269	0.297	0.248	17.1%
J Lee	0.366 (0.038)	0.089	0.262	0.256	0.231	**0.387**	0.134	0.181	0.142	0.189	0.205	-5.4%

4.4.4 Comparison among Various Name Disambiguation Methods

Table 4.3 and Table 4.4 show the performance comparison of name disambiguation between our proposed method and other competing methods for all 20 name references (one table for ArnetMiner names, and the other for CiteSeerX names). In both tables, the rows correspond to the name references and the columns (2 to 12) stand for various methods. The competing methods are grouped logically. The first group includes the baseline methods that we have designed such as random predictor (Rand) and methods using low-dimensional factorization of author-list for clustering. The second group includes various state-of-the-art network embedding methodologies, and the third group includes two methods using graphlet based graph kernels. The cell values are the performance of a method using Macro-F1 score for disambiguation of documents under a given name reference. The last column shows the overall improvement of our proposed method compared with the best competing method. Since SGD based optimization technique in our proposed embedding model is a randomized method, for each name reference we execute the method 10 times and report the average Macro-F1 score. For our method, we also show the standard deviation in the parenthesis. [5] For better visual comparison, we highlight the best Macro-F1 score of each name reference with bold-face font.

[5]Standard deviation for other competing methods are not shown due to the space limit.

As we observe, our proposed embedding model performs the best for 9 and 8 name references (out of 10) in Table 4.3, and Table 4.4, respectively. Besides, the overall percentage improvement that our method delivers over the second best method is relatively large. For an example, consider the name "S Lee" shown in the last row of Table 4.4. This is a difficult disambiguation task; from Table 4.2, it has 1091 documents and 74 distinct real-life authors ! A random predictor (Rand) obtains a Macro-F1 of only 0.057 due to the large number of classes. Whereas our method achieves 0.624 Macro-F1 score for this name reference; the second best method for this name (GF) achieves only 0.345, indicating a substantial improvement by our method. The relatively good performance of our proposed method across all the name references is due to the fact that the method is able to learn document embedding, which is particularly suited for the name disambiguation task by facilitating information exchange among the three networks (see Section 4.2).

Among the competing methods, AuthorList based methods perform poorly because the binary features are not intelligent enough to disambiguate documents, even after using traditional low dimensional embedding by non-negative matrix factorization. Graph kernel based methods such as GL3 and GL4 also have similar fate; the possible reason could be that the size-3 and size-4 graphlet structures are not decisive patterns to distinguish documents authored by different persons. On the other hand, embedding based methods are much better as they are able to learn effective features, which bring the documents authored by the same real-life person in close proximity in the feature space. This finding justifies our approach of choosing a document embedding method for solving name disambiguation. Among the competing network embedding based approaches, as we can observe from all name references, no single method emerges as a clear winner. To be more precise, PTE performs poorly as it fails to incorporate linked structural information among the documents. Both GF and LINE outperform DeepWalk in the majority of name references. This is because DeepWalk ignores the weights of the edges, which is considered to be very important in the linked document network. However, neither of embedding based competing

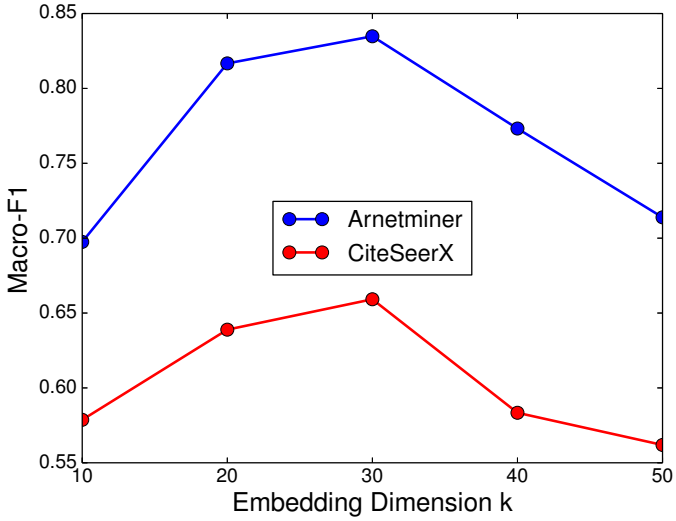


Fig. 4.2.: The effects of embedding dimension on the name disambiguation performance

methods could encode the document co-occurrence by exploiting the information from multiple networks, which is exploited by our proposed model. Besides, as mentioned earlier, our similarity ranking based objective function is better suited than the K-L divergence based objective functions for placing the nodes in the embedding space for facilitating a downstream clustering task. This is possibly a significant reason for our method to show superior performance over the existing network embedding based methods.

4.4.5 Parameter Sensitivity of Embedding Dimension

We also perform experiment to show how the embedding dimension k affects the disambiguation performance of our proposed method. Specifically, we vary the number of embedding dimension k as $\{10, 20, 30, 40, 50\}$. For the sake of space, in each of the datasets, we show the average results over all the 10 name references. The disambiguation results are given in Figure 4.2. As we observe, for both datasets, as the dimension of embeddings increases, the disambiguation performance in terms of

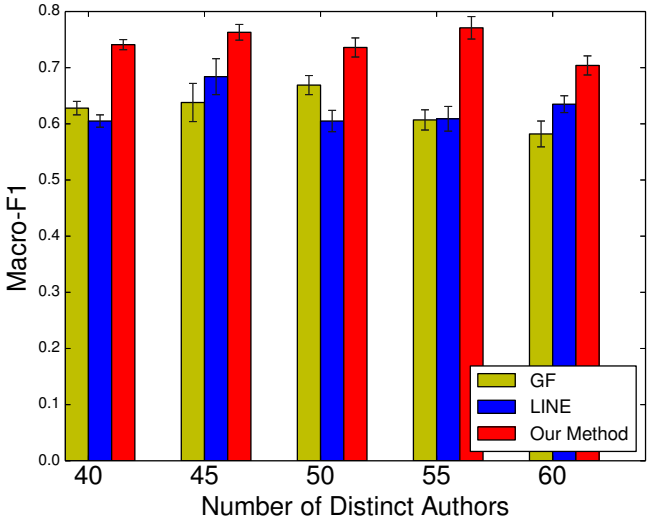


Fig. 4.3.: Macro-F1 results of multiple L values on name reference "Lei Wang" using our method, GF, and LINE (embedding dimension = 20).

Macro-F1 first increases and then decreases. The possible explanation could be that when the embedding dimension is too small, the embedding representation capability is not sufficient. However, when the embedding dimension is too large, the proposed embedding model may overfit the data, leading to the unsatisfactory disambiguation performance.

4.4.6 Performance Comparison over the Number of Clusters

One of the potential problems for name disambiguation is to determine the number of real-life persons L under a given name reference, because in real-life L is generally unknown a-priori. So a method whose performance is superior over a range of L values should be preferred. For this comparison, after learning the document representation, we use various L values as input in the HAC for name disambiguation and record the Macro-F1 score over different L for the competing methods. In our experiment, we compare the Macro-F1 value of our method with the two other best performing methods over several names. Specifically, we show this result for one name ("Lei

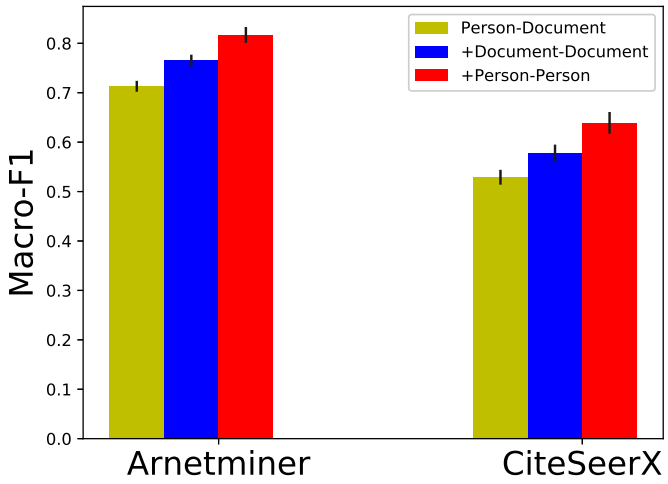
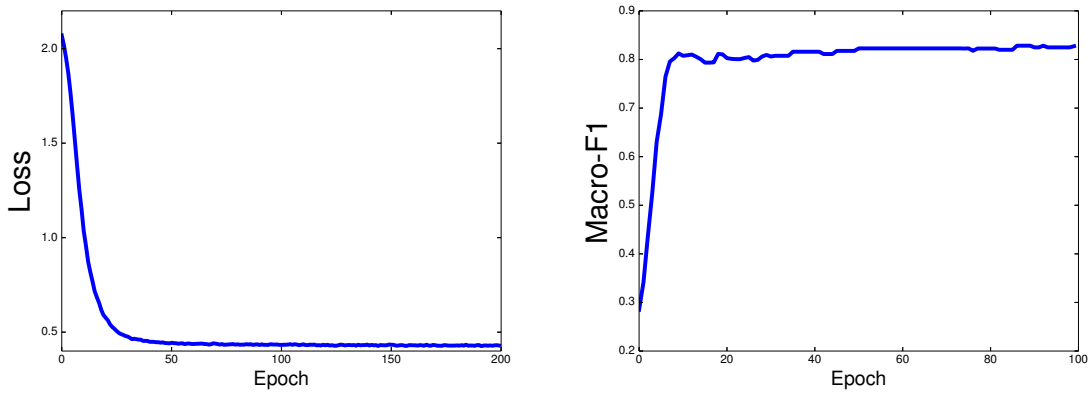


Fig. 4.4.: Component contribution analysis in terms of name disambiguation performance using Arnetminer and CiteSeerX as a whole source (embedding dimension = 20).

Wang" in Arnetminer) using bar-charts in Figure 4.3. In this figure, we compare the performance differences between our method with two other best performing methods (GF and LINE) as we vary L as $\{40, 45, 50, 55, 60\}$. Note that the actual number of distinct authors under "Lei Wang" is 48 as shown in Table 4.1. As we can see, our proposed method always outperforms the state-of-the-art with all different L values, and the overall improvement of our method over these two methods is statistically significant with a p-value of less than 0.01. Because of the robustness of our proposed embedding method for name disambiguation regardless of L values, this is a better method for the real-life application.

4.4.7 Component Contribution Analysis

Our proposed network embedding model is composed of three types of networks, namely person-person, person-document, and linked document networks. In this section we study the contribution of each of the three components for the task of name disambiguation by incrementally adding the components in the network embedding



(a) Loss vs Epoch (b) Macro-F1 vs Epoch

Fig. 4.5.: Convergence analysis in terms of both objective loss and Macro-F1 of name reference "Lei Wang" using our proposed network embedding model for name disambiguation (embedding dimension = 20).

Table 4.5.: Negative instance sampling strategy comparison in our proposed network embedding model for name disambiguation task (embedding dimension = 20). The Macro-F1 results are averaged out over 10 name references in each of the datasets.

	Arnetminer	CiteSeerX
Uniform Sampling	0.817 ± 0.015	0.639 ± 0.019
Dynamic Negative Sampling	**0.832 ± 0.011**	**0.662 ± 0.024**
Adaptive Sampling	0.809 ± 0.009	0.634 ± 0.018

model. Specifically, we first rank each individual component by its disambiguation performance in terms of Macro-F1, then add the components one by one in the order of their disambiguation power. In particular, we first add person-document graph, followed by linked document graph, and person-person graph. Figure 4.4 shows the name disambiguation performance in terms of Macro-F1 value using our proposed network embedding model with different component combinations. As we see from the figure, after adding each component, we observe improvements for both datasets, in which the results are averaged out over all the 10 name references.

4.4.8 Negative Instance Sampling Strategy Analysis

In Table 4.5, we show the name disambiguation performance in terms of Macro-F1 using various sampling techniques. As we observe, dynamic negative sampling outperforms both uniform sampling and adaptive sampling with a relative large margin. The possible explanation is due to the fact that updating the embedding vectors of "hard" negative instances generated by current prediction model makes the document vectors disambiguation-aware in the embedded space, leading to the desirable name disambiguation results. In contrast, the pre-defined exponential distribution in adaptive sampling fails to capture the true discrete distribution of negative instances produced by current embedding model, which causes the worse disambiguation performance compared to both dynamic negative sampling and uniform sampling.

4.4.9 Convergence Analysis

We further investigate the convergence of proposed network embedding algorithm shown in Section 4.3. Figure 4.5 shows the convergence analysis of our method under the name reference "Lei Wang" from Arnetminer. For each epoch, we sample $\Big(|E_{pp}| + |E_{pd}| + |E_{dd}| \Big)$ training instances to update the corresponding model embedding vectors. We can observe that our proposed network embedding approach converges approximately within 50 epochs and achieves promising convergence results on both pairwise ranking based objective loss and Macro-F1. However, as shown in Equation 4.7, the objective function in our proposed embedding model is not convex, thus reaching global optimal solution using SGD based optimization technique is a fairly challenging task. The possible remedy could be to decrease the learning rate α in SGD when number of epochs increases. Another strategy is to try multiple runs with different seeds initialization. Similar convergence patterns are observed for other name references as well.

4.5 Chapter Summary

To conclude, in this chapter we propose a novel representation learning based solution to address the name disambiguation problem. Our proposed representation learning model uses a pairwise ranking objective function which clusters the documents belonging to a single person better than other existing network embedding methods. Besides, the proposed solution uses only the relational data, so it is particularly useful for name disambiguation in anonymized network, where node attributes are not available due to the privacy concern. Our experimental results on multiple datasets show that our proposed method significantly outperforms many of the existing state-of-the-arts for name disambiguation as de-identified data.

5. BAYESIAN NON-EXHAUSTIVE CLASSIFICATION FOR ACTIVE ONLINE NAME DISAMBIGUATION

5.1 Introduction

A key limitation of most of the existing methods for name disambiguation [3, 4, 6–8, 10, 26, 32] is that they operate in a batch mode, where all records to be resolved are initially accessible to the learning algorithm and a learning model is trained using features extracted from these records. Hence, they fail to resolve emerging name ambiguities caused from the evolution of digital data, or they fail to utilize emerging evidences suggestive of merging of name entities which are separated in the existing state. Re-running a batch learning to catch up with the data evolution is not practical due to the enormity of the computation on a large digital repository. So, it is more practical to perform name entity disambiguation task in an incremental fashion by considering the streaming nature of records. We call this *online name disambiguation*, which is the focus of this chapter.

Designing an incremental, i.e., online, name disambiguation is challenging as the method must be able to adapt to a *non-exhaustive* training dataset. A training dataset is called exhaustive if it contains records for all values (classes) of the target variable, otherwise it is called non-exhaustive. In other words, it should be able to identify records belonging to new ambiguous persons who do not have any historical records in the system. After identification, the learning algorithm must re-configure the model (for instance, update the number of classes, k) so that it can correctly recover future records of this newly found ambiguous person. This is an important requirement because in real-life, for a common name, a significant number of streaming records belongs to novel (not yet seen) persons sharing that name. As an example, consider the

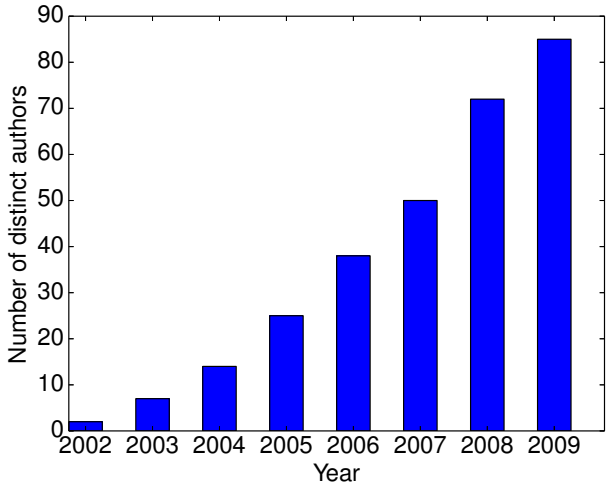


Fig. 5.1.: Name ambiguity evolution for the name "Jing Zhang"

name reference "Jing Zhang" from Arnetminer. As shown in figure 5.1, the number of distinct real-life authors in Arnetminer sharing the name "Jing Zhang" has increased from 14 to 85 between the year 2004 and 2009. Evidently, the training dataset of online name disambiguation is never exhaustive and any supervised classifier trained on the assumption of exhaustive training dataset misclassifies (with certainty) all the records belonging to a novel ambiguous person.

Besides non-exhaustiveness, *online verification* is another desirable property for an incremental name disambiguation system. Such a system asks users to provide feedback on the correctness of its prediction. Feedback collection can be automated by using online social networks or crowdsourcing platforms. As an example, consider the online digital library platform ResearchGate; it performs author name disambiguation by asking a potential researcher whether he is the author of a paper before adding that paper to that person's profile. Human feedback significantly improves the accuracy of a name disambiguation task; however, to reduce human effort the system should consult the human as infrequently as possible, and the consultation should be made for documents, for which the human feedback would yield the maximum utility for

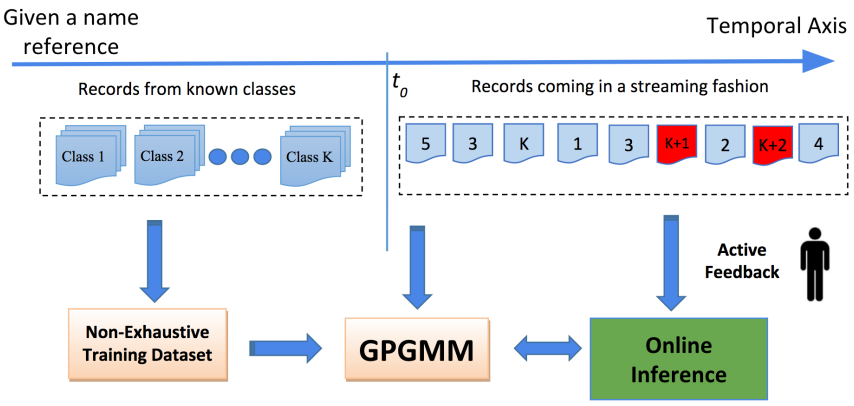


Fig. 5.2.: Bayesian non-exhaustive classification framework for active online name disambiguation

reconfiguring the model. Thus, designing an active name disambiguation system that can accommodate streaming non-exhaustive data is another focus of this work.

A few works that perform online name disambiguation by considering non-exhaustive training dataset have emerged recently. For example, Khabsa et al. [39] propose a DBSCAN based density estimation model to classify new records while they are being added incrementally. Qian et al. [38] present a probabilistic model to determine the class membership of a newly added record. Ariano et al. [40] introduce an association rule based approach for detecting unseen authors. Even though all these studies are able to adapt to the non-exhaustive scenario, their corresponding online prediction models use heuristically chosen threshold values to decide whether a record belongs to a new ambiguous author or not; such approaches are highly susceptible to the choice of threshold parameters.

In this work we propose a new Bayesian non-exhaustive classification framework for active online name disambiguation problem. Our method, which is illustrated in Figure 5.2, uses a Dirichlet Process Gaussian Mixture Model (DPGMM) as the core engine. The DPGMM facilitates online non-exhaustive classification by being partially-observed, where existing classes are modeled by observed components and emerging/future classes by the unobserved ones. The hyperparameters of the base distribution of Dirichlet process, which is chosen as a bivariate Normal $\times$ In-

verted Wishart (NIW), enables information sharing across existing as well as emerging classes. The hyperparameters are estimated using offline records initially accessible in the training set. For prediction (i.e., assigning class label to a test record), we propose two independent online inference mechanisms, the first is based on one-pass Gibbs sampler and the second is based on particle filtering. For both cases, the inference process jointly tackles online classification and emerging class discovery i.e., the inference process evaluates the probability of assigning a future record to an emerging class or to one of the existing ones. It also uses predicted label information to update the hyperparameters of our online model and thus adapt the new classification model to classify subsequent records. We also extend our method to active online name disambiguation task where the method systematically selects a small number of records and seeks user feedback regarding their true class of origin in an effort to effectively reconfigure the model.

Below we summarize the contributions of this chapter:

1. We study online name disambiguation problem in a non-exhaustive streaming setting and propose a self-adjusting Bayesian non-exhaustive model that is capable of performing online classification, and novel class discovery at the same time. To the best of our knowledge, our work is the first one to adapt Bayesian non-exhaustive classification for online name disambiguation task.

2. We propose two online inference algorithms, namely one-pass Gibbs sampler and particle filtering, for Dirichlet Process Gaussian Mixture Model to perform online non-exhaustive classification in order to efficiently evaluate the class assignment of an online record.

3. We enhance our proposed online name disambiguation approach by making it interactive, so that user guidance can be incorporated to improve the disambiguation performance. To the best of our knowledge, our model is the first work where active learning is coupled with non-exhaustive online learning for name disambiguation.

4. We use bibliographic datasets to evaluate the proposed approach against several benchmarks. The results demonstrate the superiority of the proposed approach over the state-of-the-art in online name disambiguation.

5.2 Online Name Disambiguation Challenges

For a given name reference a, assume R_n is a stream of records associated with a. The subscript n represents the identifier of the last record in the stream and the value of this identifier increases as new records are observed in the stream. Each record $r_i \in R_n$ can be represented by a d-dimensional vector which is the feature representation of the record in a metric space. In real-life, the name reference a is associated with multiple persons (say k) all sharing the same name, a. The task of name disambiguation is to partition R_n into k disjoint sets such that each partition contains records of a unique person entity. When k is fixed and known a priori, name disambiguation can be solved as a k-class classification task using supervised learning methodologies. However, for many domains the number of classes (k) is not known, rather with new records being inserted in the stream R_n , the number of distinct person entities associated with a may increase. The objective of online name disambiguation is to learn a model that assigns each incoming record into an appropriate partition containing records of a unique person entity.

Online name disambiguation is marred by several challenges, which we discuss below:

First, for a given record stream $R_n = \{r_1, \cdots, r_i, \cdots, r_n\}$, the record r_i is classified with the records leading up to r_{i-1}, i.e. R_{i-1} is our training data for this classification task. However, the record r_i may belong to a new person entity (having name a) with no previous records in R_{i-1}. This happens because for online setting, the number of real-life name entities in R_n is not fixed, rather it increases over the time. A traditional k-class supervised classification model which is trained with records of

known entities mis-classifies the new emerging record with certainty, leading to an ill-defined classification problem. So, for online name disambiguation, a learning model is needed which works in non-exhaustive setting, where instances of some classes are not at all available in the training data. In existing works, this challenge is resolved using clustering framework where a new cluster is introduced for the emerging record of a new person entity, but this solution is not robust because small changes in clustering parameters make widely varying clustering outcomes.

The second challenge is that online name disambiguation, more often, leads to a severely imbalanced classification task. This is due to the fact that in most of the real-life name disambiguation problems, the size of the true partitions of the record set R_n follows a power-law distribution. In other words, there are a few persons (dominant entities) with the name reference a to whom the majority of the records belong. Only a few records (typically one or two) belong to each of the remaining entities (with name reference a). Typically, the persons whose records appear at earlier time are dominant entities, which makes identifying novel entities an even more challenging task.

The third challenge in online name disambiguation is related to the online learning scenario, where the incoming record is not merely a test instance of typical supervised learning. Rather, the learning algorithm requires detecting whether the incoming record belongs to a novel entity, and if so, the algorithm must adapt itself and configure model to identify future records of this novel entity. Overall, this requires a self-adjusting model that updates the number of classes to accurately classify incoming records to both new and existing classes.

The final challenge in our list is related to temporal ordering of the records. In traditional classification, records do not have any temporal connotation, so an arbitrary train/test split is permitted. But, for online setting the model must respect time order of the records, i.e., a future record cannot be used for building a training model that classifies older records.

Our proposed model overcomes all the above challenges by using a principled approach.

5.3 Online Name Disambiguation on Bibliographic Data

As we have mentioned earlier, name disambiguation is a severe issue in the digital library domain. In many other domains, solving name disambiguation is easier as the method may have access to personalized attributes of an entity, such as institution affiliation, and email address. But, in digital library, the reference of a paper only includes paper title, author name, publication venue, and year of publication, which are not sufficient for disambiguation of most of the name references. Besides, in many citations the first name of the authors are often replaced by initials, which worsen the disambiguation issue. As a result, nearly, all the online bibliographic repositories, including DBLP, Google scholar, ArnetMiner, and PubMed, suffer from this issue. Nevertheless, these repositories provide timely update of the publication data along with their chronological orders, so they provide an ideal setting for evaluating the effectiveness of an online name disambiguation method.

In this work, we use bibliographic data as a case study for online name disambiguation. For each name reference a, we build a distinct classification model. The record stream R_n for the name reference a is the chronologically ordered stream of scholarly publications where a is one of the authors. To build a feature vector for a paper in R_n we extract features from its author-list, keywords from its paper title, and paper venue (journal/conference). We provide more details of feature construction in the following subsection.

5.3.1 Feature Matrix Construction and Preprocessing

For a given name reference a, say we have a record stream containing n papers for which the name reference a is in the author-list. We represent each paper with a d dimensional feature vector. Then we define a data matrix for a, in which each row represents a record and each column corresponds to a feature. In addition, each record has a class label l_i to represent the i-th distinct person sharing the name reference a. The name disambiguation task is to partition R_n into k disjoint sets such that each partition contains records of a unique person with name reference a. Note that, the k value is not fixed, rather it can increase as emerging records appear in the stream R_n.

Following earlier works on name disambiguation in the bibliographic domain [8–10], we use coauthor information, publication and venue titles as features for a publication r_i. For coauthor information, we first partition the coauthor list of each paper (except a) into authors, then define a binary feature for each author (indicating existence or not). Paper titles are processed using standard NLP tools to remove all numbers, special characters as well as stop words. After that, we generate a binary value for each of the remaining words in the title as its feature value. Paper venue (name of journal or conference) is also a binary feature, getting a value of 0 or 1 depending on whether the paper is published in a venue or not.

To address the sparsity of the generated binary feature representation, we preprocess the data by using an incremental version of non-negative matrix factorization (INNMF), which sequentially embeds the original feature matrix into a low dimensional space denoted as $X_n \in \mathbb{R}^{n \times h}$, where h is the latent dimension. Specifically, we first perform NNMF [108] in the batch mode using the initially available training records. Then for each online record, we represent it as a linear combination of a set of basis vectors generated from the training set. The coefficients serve as latent features for each online record. In order to learn the coefficients, we solve a constrained

quadratic programming problem by minimizing a least square loss function under the constraint that each coefficient is non-negative. The justification of using INNMF is to discover effective latent feature representation for each online record to better fit our proposed Normal $\times$ Normal $\times$ Invert Wishart (NNIW) data model. Considering the above feature representation of the records, in subsequent discussion we will use X_n to represent the records in R_n.

5.3.2 Problem Formulation

The active online name disambiguation problem is formally defined as follows: given a temporal partition t_0, we consider two types of records. The first type consists of a collection of n records $X_n = \{x_i\}_{i=1}^n$ whose time-stamp is smaller or equal to t_0. They serve as the training set with known labels denoted as $Y_n = \{y_i\}_{i=1}^n$, where $y_i \in \{l_1, ..., l_k\}$, and k is the number of distinct classes in the training set. The second type of records has time-stamp higher than t_0; they are represented as $\tilde{X}_{n^u} = \{\tilde{x}_i\}_{i=1}^{n^u}$, n^u (u stands for unobserved) is the number of records sequentially observed online. The online name disambiguation task is to predict the labels of these records denoted as $\tilde{Y}_{n^u} = \{\tilde{y}_i\}_{i=1}^{n^u}$, where $\tilde{y}_i \in \{l_1, ..., l_{k+\tilde{k}_{n^u}}\}$ and $\tilde{k}_{n^u}$ is the number of emerging classes associated with n^u online records.

Given an online record $\tilde{x}_i$, our proposed model computes its probability for belonging to one of the existing classes or an emerging one. Based on the computed probability, if the disambiguation result is uncertain, we request the ground-truth label information of this particular online record from the user and then re-configure the model for classifying subsequent records. Note that, user interactiveness is an added feature independent of prediction method. If the user feedback is unavailable, the method simply predicts the label of a record based on its computed probability and proceeds thereon.

5.4 Methodology

In this section we discuss our proposed Bayesian non-exhaustive online name disambiguation methodologies. The methodologies discussed in this section are domain neutral and can be applied to any domain, once an appropriately constructed feature matrix is obtained.

5.4.1 Dirichlet Process Gaussian Mixture Model

The Dirichlet Process (DP) [109] is one of the most widely used Bayesian non-parametric priors, parameterized by a concentration parameter $\alpha > 0$ and a base distribution H over a given space $\theta \in \Theta$. Although the base distribution H can be continuous, a sample $G \sim DP(\alpha, H)$ drawn from a DP is a discrete distribution. In order to represent samples G drawn from a DP, it is a common practice to use stick breaking construction [110] as below:

$$\begin{aligned} \phi_i &\sim H \\ \beta_i &\sim Beta(1, \alpha) \\ \pi_i &= \beta_i \prod_{j=1}^{i-1}(1 - \beta_j) \end{aligned} \tag{5.1}$$

As shown in Equation 5.1, in order to simulate the process of stick breaking construction, imagine we have a stick of length 1 to represent total probability. We first generate each point ϕ_i from base distribution H, which originates from our proposed Normal $\times$ Invert Wishart data model. Then we sample a random variable β_i from Beta(1, α) distribution. After that we break off a fraction β_i of the remaining stick as the weight of parameter ϕ_i, denoted as π_i. In this way it allows us to represent

random discrete probability measure G as a probability mass function in terms of infinitely many $\phi_1, ..., \phi_\infty$ and their corresponding weights $\pi_1, ..., \pi_\infty$ yielding $G = \sum_{i=1}^{\infty} \pi_i \delta_{\phi_i}$, where δ_{ϕ_i} is the point mass of ϕ_i.

Thanks to the discrete nature of G, DP offers an online clustering/classification of the streaming records as a by-product. Specifically, given a set of n records $X_n = \{x_i\}_{i=1}^n$ parameterized by $\Theta_n = \{\theta_i\}_{i=1}^n$ drawn from G, n records can be classified into k classes based on how they (the records) share parameters in Θ_n. Then, by using the definition of Chinese Restaurant Process (CRP) [109], the class label y_{n+1} for a new record x_{n+1} can be predicted as follows:

$$\begin{aligned} P(y_{n+1} = l_j | Y_n) &\propto \frac{n_j}{\alpha + n} \\ P(y_{n+1} = l_{k+1} | Y_n) &\propto \frac{\alpha}{\alpha + n} \end{aligned} \tag{5.2}$$

where l_j is one of the labels for existing classes, $j \in \{1, ..., k\}$, and l_{k+1} denotes the label of a new class. According to CRP, the probability of assigning a new incoming record to an existing class l_j is proportional to the size of that class n_j, and the probability of generating an emerging class is proportional to the concentration parameter α.

A DP Gaussian mixture model is obtained when each record $x_i \in \mathbb{R}^h$ is generated from a Gaussian distribution whose parameter $\theta_i = \{\mu_i, \Sigma_i\}$ is drawn from G. Note that we assume our collected streaming records generated by INNMF step has the property of unimodality. Thus, we use a normally distributed data model, which can model unimodal class distributions fairly well. Next, we present the Dirichlet Process Gaussian Mixture Model (DPGMM) as below:

$$
\begin{aligned}
x_i &\sim N(x_i|\theta_i) \\
\theta_i = \{\mu_i, \Sigma_i\} &\sim G \\
G &\sim DP(\alpha, H)
\end{aligned} \tag{5.3}
$$

In DPGMM, each class component is modeled using a single Gaussian distribution. Due to the discreteness of the distribution G, records sharing the same parameter θ are considered as belonging to the same class. The base distribution H in DPGMM is a conjugate Normal $\times$ Invert Wishart (NIW) prior, which is defined as follows:

$$
\begin{aligned}
H &= \text{NIW}(\mu_0, \Sigma_0, \kappa, m) \\
&= \mathcal{N}(\mu|\mu_0, \frac{\Sigma}{\kappa}) \times W^{-1}(\Sigma|\Sigma_0, m)
\end{aligned} \tag{5.4}
$$

where μ_0 is the prior mean and κ is a scaling constant that controls the deviation of the class conditional mean vectors from the prior mean. The parameter Σ_0 is a positive definite matrix that encodes our prior belief about the expected Σ. The parameter m is a scalar that is negatively correlated with the degrees of freedom. For a given record x_i, by integrating out its corresponding parameters μ_i and Σ_i, its posterior predictive distribution for a Gaussian data model and NIW prior can be obtained in the form of multivariate student-t distribution [111]:

$$
\begin{aligned}
p(x_i|y_i = l_j) &= T(x_i|\overline{\mu}_j, \overline{\Sigma}_j, \overline{v}_j) \\
\overline{\mu}_j &= \tfrac{n_j\mu_j + \kappa\mu_0}{n_j + \kappa} \\
\overline{\Sigma}_j &= \tfrac{n_j+\kappa+1}{(n_j+\kappa)(n_j+m+1-h)} \left(\Sigma_0 + (n_j - 1)S_j + \tfrac{n_j\kappa}{n_j+\kappa}(\mu_0 - \mu_j)(\mu_0 - \mu_j)^T\right) \\
\overline{v}_j &= n_j + m + 1 - h
\end{aligned} \tag{5.5}
$$

where μ_j and S_j are sample mean and sample covariance matrix for the class l_j. $\overline{\mu}_j$ is a $h \times 1$ mean vector, $\overline{\Sigma}_j$ is a $h \times h$ scale matrix, and $\overline{v}_j$ is the degree of freedom of the obtained multivariate student-t distribution.

Algorithm 3 One-Pass Gibbs Sampler for Online Name Disambiguation

Input: X_n, Y_n, n_u
Output: Final label prediction set $\hat{Y}_{pred}$
1: Initialize $\tilde{Y}_0 = \emptyset$
2: **for** i = 1 to n_u **do**
3: $\quad \tilde{y}_i \sim p(\tilde{y}_i|\tilde{Y}_{i-1}, \tilde{X}_i, Y_n, X_n)$
4: $\quad \tilde{Y}_i \leftarrow \tilde{Y}_{i-1} \cup \{\tilde{y}_i\}$
5: $\quad \hat{Y}_{pred} \leftarrow \hat{Y}_{pred} \cup \{\tilde{y}_i\}$
6: **end for**
7: **return** $\hat{Y}_{pred}$

5.4.2 Online Inference by One-Pass Gibbs Sampler

Given X_n (initially available records in vector representation using NNMF), Y_n (known labels of records in X_n), $\tilde{X}_{i-1}$ (the first $(i-1)$ records observed online), and $\tilde{Y}_{i-1}$ (the predicted labels of records in $\tilde{X}_{i-1}$), our goal is to evaluate the conditional posterior probability of class indicator variable of i'th online record $\tilde{x}_i$ as soon as the record appears online. If $\tilde{y}_i$ is the class indicator variable of $\tilde{x}_i$, the conditional posterior probability of $\tilde{y}_i$ can be derived using one-pass Gibbs sampler as below:

$$
\begin{aligned}
& p(\tilde{y}_i = l_j|\tilde{Y}_{i-1}, \tilde{X}_i, Y_n, X_n) \\
\propto & \begin{cases} \frac{n_j}{\alpha+n+i-1} T(\tilde{x}_i|\overline{\mu}_j, \overline{\Sigma}_j, \overline{v}_j) \text{ if } j \in \{1, ..., k+\tilde{k}_{i-1}\} \\ \frac{\alpha}{\alpha+n+i-1} T(\tilde{x}_i), \text{ if } j = k+\tilde{k}_{i-1}+1 \end{cases}
\end{aligned} \tag{5.6}
$$

From Equation 5.6, the conditional posterior probability of $\tilde{y}_i$ depends on the posterior predictive likelihood of $\tilde{x}_i$ in the form of multivariate student-t distribu-

tion and CRP prior of the corresponding class component. Specifically, the incoming online record $\tilde{x}_i$ belongs to one of the existing classes with probability proportional to $\frac{n_j}{\alpha+n+i-1}T(\tilde{x}_i|\overline{\mu}_j, \overline{\Sigma}_j, \overline{v}_j)$, and a new class with probability proportional to $\frac{\alpha}{\alpha+n+i-1}T(\tilde{x}_i)$. Note that $T(\tilde{x}_i)$ is another multivariate student-t distribution by setting all sufficient statistics in Equation 5.5 to empty sets.

The pseudo-code of the proposed one-pass Gibbs sampler for online name disambiguation is summarized in Algorithm 3. Given a collection of n records initially available in training set X_n, and their corresponding true label information Y_n, we aim to predict the class indicator variables of n_u records sequentially observed online. Specifically, from line 2-6, we utilize the conditional posterior probability shown in Equation 5.6 to decide the class assignment of each online record, denoted as $\tilde{y}_i$. After processing all n_u online records, we return the predicted class set $\hat{Y}_{pred}$ for final evaluation in line 7.

5.4.3 Online Inference by Particle Filtering

For the one-pass Gibbs sampler, the accumulated classification error from all mislabeled online records is propagated and eventually the model is likely to diverge from its true posterior distribution leading to poor online disambiguation performance. To address this issue, we develop a Sequential Importance Sampling with Resampling (SISR) [112] technique, also known as particle filtering in the literature. In contrast to a one-pass Gibbs sampler, particle filtering employs a set of particles, whose weights can be incrementally updated as new records appear online. Each particle maintains class configurations of all observed online records and the weight of a particle indicates how well the particle fits the data. Resampling ensures that particles with high weights are more likely to be replicated and the ones with low weights are more likely to be eliminated. By keeping a diverse set of class configurations, particle filtering

allows for more effective exploration of the state-space and often generates a better local optimum than that would be obtained by a one-pass Gibbs sampler.

Specifically, in the particle filtering framework, for each online record, we approximate its true class posterior distribution by a discrete distribution defined by a set of particles and their weights, which can be incrementally updated without having access to all past records. Mathematically, we are interested in predicting $\tilde{Y}_i$, i.e., the class labels for all $\tilde{X}_i$ at the time $\tilde{x}_i$ appears online. The prediction can be done by finding the expectation of the posterior distribution of class indicator variables, namely $E_{p(\tilde{Y}_i|\tilde{Y}_{i-1},\tilde{X}_i,Y_n,X_n)}\left[\tilde{Y}_i\right]$. By using an importance function $q(\tilde{Y}_i|\tilde{Y}_{i-1},\tilde{X}_i,Y_n,X_n)$ to sample particles, in which case the $E_{p(\tilde{Y}_i|\tilde{Y}_{i-1},\tilde{X}_i,Y_n,X_n)}\left[\tilde{Y}_i\right]$ can be approximated as below:

$$\begin{aligned} & E_{p(\tilde{Y}_i|\tilde{Y}_{i-1},\tilde{X}_i,Y_n,X_n)}\left[\tilde{Y}_i\right] \\ = & \int \tilde{Y}_i p(\tilde{Y}_i|\tilde{Y}_{i-1},\tilde{X}_i,Y_n,X_n)d\tilde{Y}_i \\ = & \int \tilde{Y}_i W_i\Big(\tilde{Y}_i\Big) q(\tilde{Y}_i|\tilde{Y}_{i-1},\tilde{X}_i,Y_n,X_n)d\tilde{Y}_i \\ \approx & \sum_{m=1}^{M} \tilde{Y}_i^m W_i\Big(\tilde{Y}_i^m\Big)\delta_{\tilde{Y}_i^m} \end{aligned} \tag{5.7}$$

where M is the number of particles, $\tilde{Y}_i^m$ represents the class configurations of first i online records in m-th particle, and $W_i\Big(\tilde{Y}_i^m\Big) = \frac{p(\tilde{Y}_i^m|\tilde{Y}_{i-1}^m,\tilde{X}_i,Y_n,X_n)}{q(\tilde{Y}_i^m|\tilde{Y}_{i-1}^m,\tilde{X}_i,Y_n,X_n)}$ is the corresponding weight of the m-th particle when i-th online record is observed. Using the chain rule, the particle weights can be sequentially updated as follows:

$$\begin{aligned} & W_i\Big(\tilde{Y}_i^m\Big) = \frac{p(\tilde{Y}_i^m|\tilde{Y}_{i-1}^m,\tilde{X}_i,Y_n,X_n)}{q(\tilde{Y}_i^m|\tilde{Y}_{i-1}^m,\tilde{X}_i,Y_n,X_n)} \\ = & W_{i-1}\Big(\tilde{Y}_{i-1}^m\Big)\frac{p(\tilde{x}_i|\tilde{Y}_i^m,\tilde{X}_{i-1},Y_n,X_n)p(\tilde{y}_i|\tilde{Y}_{i-1}^m,Y_n)}{p(\tilde{x}_i|\tilde{Y}_{i-1}^m,\tilde{X}_{i-1},Y_n,X_n)q(\tilde{y}_i|\tilde{Y}_{i-1}^m,\tilde{X}_i,Y_n,X_n)} \end{aligned} \tag{5.8}$$

To further simplify the formula, we set importance function to be CRP prior of the class indicator variable $\tilde{y}_i$, namely $q(\tilde{y}_i|\tilde{Y}_{i-1}^m,\tilde{X}_i,Y_n,X_n) = p(\tilde{y}_i|\tilde{Y}_{i-1}^m,Y_n)$. Further-

more, $p(\tilde{x}_i|\tilde{Y}_{i-1}^m, \tilde{X}_{i-1}, Y_n, X_n)$ is constant with respect to $\tilde{Y}_i^m$. Thus the weight update formula in Equation 5.8 can be simplified as below:

$$W_i\left(\tilde{Y}_i^m\right) \propto W_{i-1}\left(\tilde{Y}_{i-1}^m\right) p(\tilde{x}_i|\tilde{Y}_i^m, \tilde{X}_{i-1}, Y_n, X_n) \tag{5.9}$$

In Equation 5.9, $p(\tilde{x}_i|\tilde{Y}_i^m, \tilde{X}_{i-1}, Y_n, X_n)$ is a multivariate student-t distribution under the DPGMM. Thus at the time online record $\tilde{x}_i$ appears, the weights of all M particles can be updated and then normalized, namely $W_i\left(\tilde{Y}_i^m\right) = \frac{W_i(\tilde{Y}_i^m)}{\sum_{p=1}^{M} W_i\left(\tilde{Y}_i^p\right)}$, into a discrete probability distribution to approximate the true conditional posterior distribution of its class indicator variable $\tilde{y}_i$.

In order to alleviate the problem of particle degeneracy and avoid the situation that all but a few particle weights are close to zero, we add a stratified resampling step as suggested in [112]. The general philosophy of this resampling step is to replicate particles with high weights and eliminate particles with low weights. At the time $\tilde{x}_i$ appears online, we use the weight update formula 5.9 to compute weights of all particles. Then we calculate an estimate of the effective number of particles, which is defined as $ENP = \frac{1}{\sum_{m=1}^{M}[W_i(\tilde{Y}_i^m)]^2}$. If this value is less than a given threshold, we perform resampling. Specifically, we draw M particles from the current particle set (with replacement) with probabilities proportional to their weights and then we replace the current particle set with the new one. Meanwhile, we reset the weights of all particles to a uniform weight, which is $\frac{1}{M}$. In summary, in the particle filtering framework, after sampling particles using the CRP prior and updating particle weights as in Equation 5.9, we either retain weighted particles, in which case the weights are accumulated over time, or we resample particles so that they have uniform weights.

The pseudo-code of the proposed particle filtering algorithm for online name disambiguation is summarized in Algorithm 4. Specifically, in line 1, we initialize the class configurations of all particles as $\emptyset$ and their weights as uniform weights. From line 2-8, we perform particle sampling and weight update steps. In line 9, if the effective number of particles criteria ENP is below a threshold ENP_{thr}, we perform

stratified resampling as shown in line 10-14. For the final prediction of class indicator variable for online record $\tilde{x}_i$, we sum the particle weights based on the particle class labels assigned to $\tilde{x}_i$ and choose the class label with the maximum weights as the final class label, denoted as $\hat{y}_i$ in line 15. Finally, we return the predicted class set $\hat{Y}_{pred}$ for evaluation. In contrast to particle filtering in Algorithm 4, one-pass Gibbs sampler shown in Algorithm 3 can be approximately considered as a special but very restricted case of particle filtering with only one particle by setting the number of particles $M = 1$.

Algorithm 4 Particle Filtering Algorithm for Online Name Disambiguation

Input: X_n, Y_n, n_u, M, ENP_{thr}
Output: Final label prediction set $\hat{Y}_{pred}$

1: Initialize $\tilde{Y}_0^m = \emptyset$ and $W_0(\tilde{Y}_0^m) = \frac{1}{M}$ for all $m \in \{1, 2, ..., M\}$
2: **for** i = 1 to n_u **do**
3: **for** m = 1 to M **do**
4: $\tilde{y}_i \sim p(\tilde{y}_i|\tilde{Y}_{i-1}^m, Y_n)$
5: $\tilde{Y}_i^m \leftarrow \tilde{Y}_{i-1}^m \cup \{\tilde{y}_i\}$
6: $W_i\left(\tilde{Y}_i^m\right) \propto W_{i-1}\left(\tilde{Y}_{i-1}^m\right) p(\tilde{x}_i|\tilde{Y}_i^m, \tilde{X}_{i-1}, Y_n, X_n)$
7: **end for**
8: Normalize particle weights
9: **if** $ENP \leq ENP_{thr}$ **then**
10: **for** m = 1 to M **do**
11: Resample $\tilde{Y}_i^m$ with probability $\propto W_i\left(\tilde{Y}_i^m\right)$
12: $W_i\left(\tilde{Y}_i^m\right) = \frac{1}{M}$
13: **end for**
14: **end if**
15: Aggregate the particle weights based on all M particle class labels and predict the label of $\tilde{x}_i$ with maximum weights, namely $\hat{Y}_{pred} \leftarrow \hat{Y}_{pred} \cup \{\hat{y}_i\}$
16: **end for**
17: **return** $\hat{Y}_{pred}$

5.5 Active Online Name Disambiguation

Based on our developed particle filtering based online inference algorithm, we propose an active learning framework for online name disambiguation, which mainly consists of the two steps as below:

Active Selection: The objective of this step is to identify records with the most uncertain disambiguation results based on the posterior probability and seek user feedback for these cases for true label information. Specifically, for each online record, we first estimate its class conditional posterior probability using particle filtering (Section 5.4.3), then we use an entropy-based criteria to quantify the confidence of the tentative disambiguation result. Let the probability of an online record belonging to class l_j be p_j, and total number of predicted classes among all particle configurations be $|J|$. Then the entropy can be calculated as $-\sum_{j=1}^{|J|} p_j log p_j$. Note that the range of computed entropy values is between 0 and $log|J|$, where 0 means the disambiguation prediction is most confident and $log|J|$ means the disambiguation prediction is least confident. If the entropy is larger than a user-defined threshold, i.e., $\tau * log|J|$ ($0 \leq \tau \leq 1$), we consider the disambiguation result as uncertain and seek user feedback to obtain true class assignment for this particular record.

Model Update: The goal of this step is to refine the conditional posterior probability of class indicator variable of current online record when its true label is offered by users. Specifically, we first use this ground-truth label information to update class configurations of all particles with respect to this particular online record and then refine the weight of each particle using Equation 5.9.

5.6 Experimental Results

In the experiment, we consider bibliographic data in a temporal stream format and disambiguate authors by partitioning their records (papers) into homogeneous groups. Specifically, we compare our proposed Bayesian non-exhaustive classification

Table 5.1.: Arnetminer name disambiguation dataset

Name Reference	# Records	# Attributes	# Distinct Authors
Kai Zhang	66	488	24
Bo Liu	124	749	47
Jing Zhang	231	1456	85
Yong Chen	84	551	25
Yu Zhang	235	1440	72
Hao Wang	178	1074	48
Wei Xu	153	1037	48
Lei Wang	308	1819	112
Bin Li	181	1142	60
Ning Zhang	127	744	33
Feng Liu	149	919	32
Lei Chen	196	1052	40
David Brown	61	437	25
Yang Wang	195	1227	55
Gang Chen	178	1049	47
X. Zhang	62	601	40
Yun Wang	46	360	19
Z. Wang	47	498	38
Bing Liu	182	897	18
Yang Yu	71	444	19
Ji Zhang	64	398	16
Bin Yu	105	600	17
Lu Liu	58	425	17
Ke Chen	107	603	16
Gang Luo	47	270	9

framework with various existing methods to demonstrate its superiority over those methods for performing online name disambiguation. Furthermore, we also demonstrate the usages of proposed active online name disambiguation on a real-world name reference.

5.6.1 Datasets

A key challenge for the evaluation of name disambiguation task is the lack of availability of labeled datasets from diverse application domains. In recent years, the bibliographic repository site, Arnetminer has published several ambiguous author name references along with respective ground truths (paper list of each real-life person), which we use for evaluation. Specifically we use 25 highly ambiguous (having

a larger number of distinct authors for a given name) name references and show the performance of our method on these name references. The statistics of each name reference are shown in Table 5.1. In this table, we show the number of records, the number of binary attributes (explained in Section 5.3.1) and the number of distinct authors associated with that name reference. It is important to understand that the online name disambiguation model is built on a name reference, not on a source dataset, like Arnetminer as a whole, so each name reference is a distinct dataset on which the evaluation is performed.

5.6.2 Competing Methods

In order to illustrate the merit of our proposed approach, we compare our model with the following benchmark techniques. Among these the first two are existing state-of-the- art online name disambiguation methods, and the latter two are baselines that we have designed.

1. **Qian's Method [38]** Given the collection of training records initially available, for a new record, Qian's method computes class conditional probabilities for existing classes. This approach assumes that all the attributes are independent and the procedure of probability computation is based on the occurrence count of each attribute in all records of each class. Then the computed probability is compared with a pre-defined threshold value to determine whether the newly added record should be assigned to an existing class, or to a new class not yet included in the previous data.

2. **Khabsa's Method [39]** Given the collection of training records initially available this approach first computes the ϵ-neighborhood density for each online sequentially observed record. The ϵ-neighborhood density of a new record is considered as the set of records within ϵ euclidean distance from that record. Then if the neighborhood is sparse, the new record is assigned to a new class.

Otherwise, it is classified into the existing class that contains the most records in the ϵ-neighborhood of the new record.

3. **BernouNaive-HAC:** In this baseline, we first model the data with a multivariate Bernoulli distribution (features are binary, so Bernoulli distribution is used) and train a Naive Bayes classifier. This classifier returns class conditional probabilities for each record in the test set which we use as meta features in a hierarchical agglomerative clustering (HAC) framework.

4. **NNMF-SVM-HAC:** We perform NNMF on our binary feature matrix and use the coefficients returned by NNMF to train a linear SVM. Class conditional probabilities for each test record are used as meta features in a hierarchical agglomerative clustering (HAC) framework the same way described above.

5.6.3 Experimental Setting and Implementation

For each of the 25 name references, we aim to build a separate model to classify the online records belonging to existing classes represented in the training set, as well as identifying records belonging to emerging classes not represented in the training set. In particular, we first train the model using the training set initially available, then we add the records in the test set one-by-one in order to simulate new incoming streaming data. The train and test partition is based on the temporal order of each record in the dataset. To be more precise, we put the most recent T_0 years' records into the test set and the records from earlier years into the initially available training set. Furthermore, we verify how the performance of our proposed model varies as we tune the value of T_0. In the experiment, we set T_0 as 2 and 3. For the evaluation metric, we use mean-F1 measure [99], which is unweighted average of F1-measure of individual classes. The range of mean-F1 measure is between 0 and 1, and a higher value indicates better disambiguation performance.

Our proposed Bayesian non-exhaustive classification framework has a few tunable parameters. Among them, the set of prior parameters $(\Sigma_0, \mu_0, m, \kappa)$ in the base

distribution of NIW can be learned from the training set. For example, we use the mean of training set to estimate μ_0, and set Σ_0 to be the pooled covariance matrix as suggested in [113]. For m and κ, we use vague priors for fixing their values to $h+100$ and 100 respectively. In addition to that, there are three additional user-defined parameters in our proposed framework. Specifically, we set latent dimension h in INNMF as 10, concentration parameter α in Dirichlet Process as 100, and number of particles M to be 100. Finally, in particle filtering, when the effective number of particles is below $\frac{M}{2}$ as suggested by [112], we perform resampling.

For all the competing methods, we use identical set of features (before dimensionality reduction). We vary the probability threshold value of Qian's method and ϵ value of Khabsa's method by cross validation on the training dataset. and select the ones that obtain the best disambiguation performance in terms of Mean-F1 score. For BernouNaive-HAC and NNMF-SVM-HAC methods, during the hierarchical agglomerative clustering step, we tune the number of clusters in training set by cross validation in order to get the best disambiguation result.

For both data processing and model implementation, we write our own code in Python and use NLTK, NumPy, SciPy, scikit-learn, and filterpy libraries for data cleaning, linear algebra and machine learning operations. We run all the experiments on a 2.1 GHz Machine with 8GB memory running Linux operating system.

5.6.4 Performance Comparison with Competing Methods

Table 5.2 and Table 5.3 show the online name disambiguation performance between our proposed method and other competing methods for all 25 name references. In both tables, #train records and #test records columns show the number of training and test records. #emerge records column is the number of records in test set with their corresponding classes not represented in the initial training set, and #emerge classes column denotes the number of emerging classes not represented in the training set. The columns 6-11 show the performance of a method using mean-F1 score

Table 5.2.: Comparison of mean-F1 values using records with most recent 2 years as test set. Paired t-test is conducted on all performance comparisons and it shows that all improvements are significant at the 0.05 level.

Name Reference	# train records	# test records	# emerge records	# emerge classes	**Gibbs Sampler**	**Particle Filter**	BernouNaive-HAC	NNMF-SVM-HAC	Qian's Method [38]	Khabsa's Method [39]	Improv.
Kai Zhang	42	24	15	8	0.633 (0.041)	**0.661 (0.013)**	0.605	0.621	0.619	0.518	6.4%
Bo Liu	99	25	11	8	0.716 (0.033)	**0.804 (0.011)**	0.733	0.719	0.714	0.559	9.7%
Jing Zhang	121	110	56	35	0.591 (0.028)	**0.639 (0.008)**	0.554	0.566	0.590	0.631	1.3%
Yong Chen	70	14	5	5	**0.889 (0.016)**	0.807 (0.006)	0.852	0.794	0.848	0.833	4.3%
Yu Zhang	124	111	62	30	0.535 (0.013)	**0.678 (0.008)**	0.498	0.516	0.515	0.502	31.4%
Hao Wang	148	30	9	8	**0.747 (0.026)**	0.672 (0.009)	0.635	0.639	0.702	0.581	6.4%
Wei Xu	127	26	11	10	0.844 (0.033)	**0.892 (0.012)**	0.811	0.750	0.767	0.689	10.0%
Lei Wang	245	63	28	24	0.705 (0.012)	**0.722 (0.007)**	0.701	0.708	0.703	0.620	2.0%
Bin Li	154	27	11	9	0.807 (0.029)	**0.865 (0.011)**	0.775	0.733	0.775	0.743	11.6%
Feng Liu	104	45	6	5	0.589 (0.031)	**0.719 (0.022)**	0.501	0.499	0.399	0.339	43.5%
Lei Chen	96	100	24	18	0.356 (0.043)	0.438 (0.012)	**0.646**	0.527	0.430	0.222	-32.2%
Ning Zhang	97	30	16	12	0.635 (0.021)	**0.713 (0.018)**	0.669	0.685	0.647	0.608	4.1%
David Brown	48	13	4	3	0.839 (0.019)	**0.937 (0.006)**	0.904	0.593	0.816	0.450	3.7%
Yang Wang	118	77	38	20	0.469 (0.033)	**0.698 (0.009)**	0.513	0.549	0.325	0.440	27.1%
Gang Chen	113	65	20	14	**0.821 (0.004)**	0.816 (0.012)	0.474	0.467	0.451	0.357	73.2%
X. Zhang	54	8	5	5	0.969 (0.018)	**1.0 (0.011)**	0.593	0.485	0.952	0.222	5.0%
Yun Wang	31	15	6	6	0.680 (0.011)	**0.762 (0.005)**	0.512	0.479	0.644	0.358	18.3%
Z. Wang	41	6	5	4	0.884 (0.023)	**0.906 (0.003)**	0.693	0.712	0.889	0.701	1.9%
Bing Liu	156	26	4	4	0.495 (0.009)	**0.727 (0.004)**	0.318	0.466	0.356	0.406	56.0%
Yang Yu	51	20	6	6	0.503 (0.013)	0.648 (0.003)	0.499	0.523	**0.684**	0.493	-5.3%
Ji Zhang	46	18	7	5	0.512 (0.024)	**0.616 (0.008)**	0.412	0.392	0.514	0.545	13.0%
Bin Yu	87	18	7	4	0.469 (0.011)	**0.579 (0.004)**	0.488	0.526	0.564	0.540	2.7%
Lu Liu	24	34	17	9	0.406 (0.012)	**0.497 (0.009)**	0.417	0.429	0.399	0.346	15.9%
Ke Chen	70	37	7	6	0.370 (0.012)	0.439 (0.005)	0.401	0.398	0.423	**0.501**	-12.4%
Gang Luo	30	17	6	3	0.603 (0.022)	**0.865 (0.005)**	0.622	0.693	0.744	0.786	10.1%

Table 5.3.: Comparison of mean-F1 values using records with most recent 3 years as test set. Paired t-test is conducted on all performance comparisons and it shows that all improvements are significant at the 0.05 level

Name Reference	# train records	# test records	# emerging records	# emerging classes	**Gibbs Sampler**	**Particle Filter**	BernouNaive-HAC	NNMF-SVM-HAC	Qian's Method [38]	Khabsa's Method [39]	Improv.
Kai Zhang	27	39	20	10	0.602 (0.021)	**0.632 (0.011)**	0.503	0.584	0.520	0.510	8.2%
Bo Liu	66	58	29	21	0.699 (0.011)	**0.767 (0.022)**	0.612	0.606	0.612	0.631	21.6%
Jing Zhang	82	149	77	47	0.568 (0.022)	**0.601 (0.009)**	0.480	0.446	0.423	0.419	25.2%
Yong Chen	54	30	12	8	0.775 (0.047)	**0.788 (0.021)**	0.615	0.701	0.615	0.545	12.4%
Yu Zhang	87	148	71	38	0.457 (0.013)	**0.639 (0.019)**	0.445	0.615	0.447	0.412	3.9%
Hao Wang	115	63	17	12	**0.698 (0.031)**	0.545 (0.011)	0.513	0.572	0.540	0.512	22.0%
Wei Xu	101	52	17	14	0.734 (0.051)	**0.836 (0.028)**	0.683	0.603	0.635	0.586	22.4%
Lei Wang	173	135	67	45	0.693 (0.044)	**0.701 (0.031)**	0.560	0.522	0.536	0.428	25.2%
Bin Li	108	73	37	23	0.777 (0.009)	**0.828 (0.004)**	0.532	0.574	0.588	0.545	40.8%
Feng Liu	70	79	9	8	0.545(0.017)	**0.618 (0.027)**	0.488	0.527	0.379	0.424	17.3%
Lei Chen	65	131	39	25	0.332 (0.029)	0.382 (0.007)	**0.493**	0.447	0.398	0.176	-22.5%
Ning Zhang	76	51	32	19	0.589 (0.034)	0.682 (0.019)	**0.744**	0.531	0.420	0.378	-8.3%
David Brown	39	22	17	7	0.734 (0.008)	**0.899 (0.002)**	0.751	0.631	0.752	0.478	19.5%
Yang Wang	92	103	46	25	0.436 (0.012)	**0.627 (0.011)**	0.313	0.298	0.225	0.240	100.3%
Gang Chen	89	89	27	19	**0.799 (0.008)**	0.737 (0.012)	0.347	0.407	0.383	0.221	96.3%
X. Zhang	53	9	6	6	0.959 (0.016)	**0.992 (0.005)**	0.563	0.445	0.905	0.202	9.6%
Yun Wang	25	21	17	9	0.535 (0.012)	**0.668 (0.006)**	0.501	0.438	0.567	0.385	17.8%
Z. Wang	35	12	10	8	0.842 (0.013)	**0.894 (0.011)**	0.613	0.652	0.879	0.424	1.7%
Bing Liu	141	41	9	5	0.415 (0.019)	**0.648 (0.006)**	0.307	0.481	0.286	0.371	34.7%
Yang Yu	37	34	10	8	0.471 (0.014)	**0.539 (0.007)**	0.459	0.508	0.510	0.447	5.7%
Ji Zhang	41	23	8	6	0.461 (0.017)	**0.591 (0.003)**	0.402	0.349	0.494	0.483	19.6%
Bin Yu	80	25	11	47	0.430 (0.012)	**0.559 (0.008)**	0.461	0.539	0.423	0.463	3.7%
Lu Liu	10	48	34	13	0.336 (0.013)	0.409 (0.011)	0.401	0.418	0.317	**0.426**	-4.0%
Ke Chen	54	53	20	7	0.313 (0.012)	0.339 (0.009)	0.396	0.337	0.404	**0.442**	-23.3%
Gang Luo	20	27	8	4	0.627 (0.021)	**0.810 (0.009)**	0.612	0.674	0.675	0.726	11.6%

for online disambiguation of records under a given name reference. The last column represents the overall improvement of our proposed method compared with the best competing method. Since both one-pass gibbs sampler and particle filtering based

online inference techniques in our proposed online name disambiguation model are randomized algorithms, for each name reference we run the method 30 times and report the average mean-F1 score. In addition, for our method, we also show the standard deviation in the parenthesis [1]. For better visual comparison, we highlight the best mean-F1 score of each name reference with bold-face font.

If we compare the 25 datasets between the two tables, for higher T_0 value, the number of training records decreases, the number of test records, emerging records, and emerging classes increase. It makes the online name disambiguation task in the first setting (2 years test split, i.e., $T_0 = 2$) easier than the second setting ($T_0 = 3$) . This is reflected in the mean-F1 values of all the name references across both tables. For example, for the first name reference, Kai Zhang, mean-F1 score of particle filtering across these two tables are 0.661 and 0.632 respectively. This performance reduction is caused by the increasing number of emerging classes; 8 in Table 5.2, and 10 in Table 5.3. Another reason is decreasing number of training instances; 42 in Table 5.2, and 27 in Table 5.3. As can be seen in both tables, our name disambiguation dataset contains a large number of emerging records in the test data, all of these records will be misclassified with certainty by any traditional exhaustive name disambiguation methods. This is our main motivation for designing a non-exhaustive classification framework for online name disambiguation task.

Now we compare our method with the four competing methods. As we observe, our proposed online name disambiguation model performs best for 22 and 21 name references (out of 25) in Table 5.2 and Table 5.3, respectively. Besides, the overall percentage improvement that our method delivers over the second best method is relatively large. For an example, consider the name reference "Jing Zhang" shown in Table 5.3. This is a difficult online name disambiguation task as it contains a large number of emerge records in the test set (77 emerge records from 47 emerge classes), thus any traditional classifier will misclassify all these emerge records with certainty. For our proposed particle filter and one-pass Gibbs sampler, it achieves

[1]Standard deviation for other competing methods are not shown due to space limit.

0.601 and 0.568 mean-F1 score for this name reference, respectively; whereas the best competing method for this name (BernouNaive-HAC) obtains only 0.480, indicating a substantial improvement (25.2%) by our method (particle filtering). The relatively good performance of the proposed method may be due to our non-exhaustive learning methodologies. It also suggests that the base distribution used by the proposed Dirichlet process prior model whose parameters are estimated using data from known classes can be generalized for the class distributions of unknown classes as well.

When we compare between our proposed online inference approaches, particle filtering performs better than one-pass Gibbs sampler. The possible explanation could be that one-pass Gibbs sampler fails to maintain multiple local optimal solutions and prevent error propagations effectively during the online execution. In comparison, particle filtering offers more accurate approximation of class posterior distribution for each online record, which leads to better mean-F1 performance across most of name references.

In contrast, among all the competing methods, Qian's method and Khabsa's method perform the worst as they fail to incorporate prior information about class distribution into the models and the results are very sensitive to the selections of threshold parameters. On the other hand both BernouNaive-HAC and NNMF-SVM-HAC operate in an off-line framework. Although for some name references mean-F1 scores obtained by these techniques are higher than our proposed method, there is a clear trend favoring our proposed method over these methods—latter cannot explicitly identify streaming records of new ambiguous classes in an online setting.

Table 5.4 presents the result of automatic estimation of number of distinct real-life persons in test set using both one-pass Gibbs sampler and particle filtering. From the table, as we observe, for most of name references, the predicted number of distinct persons are slightly larger than the actual ones. The possible explanation could be that our name disambiguation datasets contain skewed classes, and DPGMM tends to produce multiple components for each class. Despite that, as a remark, our predicted results are very close to the actual ones, which demonstrate the effectiveness of our

Table 5.4.: Results of number of distinct real-life persons under our proposed Bayesian non-exhaustive classification framework using most recent 3 years' records as test set

Name Reference	# Actual Authors	# Predicted Authors (Gibbs Sampler)	# Predicted Authors (Particle Filter)
Kai Zhang	15	19.1 ± 3.2	20.3 ± 3.7
Bo Liu	30	34.2 ± 2.0	31.3 ± 2.9
Jing Zhang	62	51.4 ± 4.8	56.3 ± 5.6
Yong Chen	13	14.2 ± 1.9	16.9 ± 2.2
Yu Zhang	55	60.3 ± 4.8	56.8 ± 4.3
Hao Wang	27	35.6 ± 2.3	33.8 ± 3.2
Wei Xu	27	30.2 ± 1.8	29.0 ± 2.5
Lei Wang	65	70.1 ± 4.6	68.5 ± 3.4
Bin Li	32	35.6 ± 4.1	37.0 ± 3.5
Feng Liu	26	29.7 ± 3.3	23.8 ± 2.9
Lei Chen	12	18.2 ± 2.9	14.5 ± 4.1
Ning Zhang	23	21.1 ± 1.7	24.8 ± 2.2
David Brown	11	13.4 ± 3.6	14.2 ± 1.8
Yang Wang	36	28.9 ± 4.5	35.2 ± 2.6
Gang Chen	28	30.2 ± 1.8	31.8 ± 2.0
X. Zhang	9	10.3 ± 3.7	12.1 ± 2.8
Yun Wang	12	14.3 ± 3.9	11.7 ± 0.9
Z. Wang	10	12.4 ± 3.7	11.6 ± 1.2
Bing Liu	8	10.6 ± 1.9	9.2 ± 2.2
Yang Yu	14	17.4 ± 5.7	15.8 ± 2.6
Ji Zhang	11	10.3 ± 6.9	12.1 ± 2.3
Bin Yu	11	9.3 ± 2.8	12.3 ± 1.6
Lu Liu	15	12.1 ± 3.3	16.3 ± 1.5
Ke Chen	12	14.1 ± 2.4	13.2 ± 1.9
Gang Luo	5	7.9 ± 2.0	10.2 ± 2.7

proposed online name disambiguation framework for estimating the number of actual real-life persons accurately under the non-exhaustive setup.

5.6.5 Feature Contribution Analysis

We investigate the contribution of each of the defined features (coauthor, keyword, venue) for the task of online name disambiguation. Specifically, we first rank the individual features by their performance in terms of mean-F1 score, then add the features one by one in the order of their disambiguation power. In particular, we first use author-list, followed by keywords, and publication venue. In each step, we evaluate the performance of our proposed online name disambiguation method using the most recent two years' publication records as test set. Figure 5.3 shows the mean-

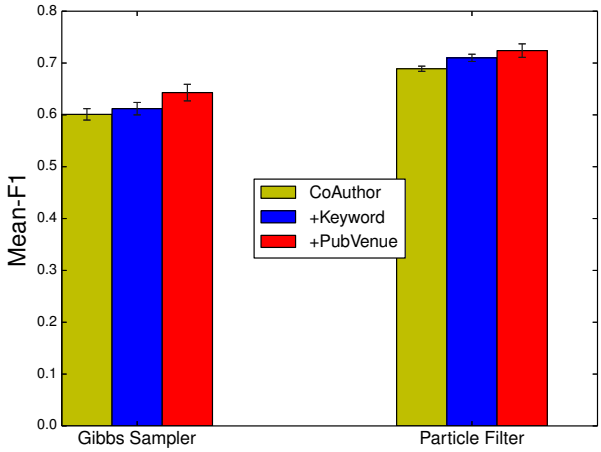
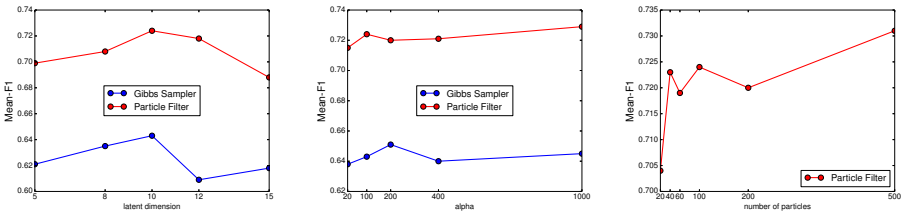


Fig. 5.3.: Feature contribution analysis using most recent 2 years' publication records as test set. The results are averaged out over all 25 name references for better visualization.

F1 value of our method with different feature combinations. As we can see from this figure, for both one-pass Gibbs sampler and particle filtering based online inference techniques, after adding each feature group we observe improvements in terms of mean-F1 score, in which the results are averaged out over all the 25 name references for better visualization.

5.6.6 Study of Running Time

A very desirable feature of our proposed Bayesian non-exhaustive classification model is its running time. For example, using the most recent two years' records as test set, on the name reference "Kai Zhang" containing 66 papers with 10 latent dimensionality, it takes around 0.29 and 1.09 seconds on average to assign the test papers to different real-life authors for one-pass Gibbs sampler and particle filtering, respectively. For the name reference "Lei Wang" with 308 papers using same number of latent dimensionality, it takes around 1.95 and 5.91 seconds on average under the same setting. This suggests only a linear increase in computational time with respect to the number of records. However in addition to number of records, the computa-



(a) Mean-F1 vs. latent dimension (b) Mean-F1 vs. concentration parameter (c) Mean-F1 vs. # particles

Fig. 5.4.: The effects of latent dimension h, concentration parameter α in Dirichlet process, and number of particles M in particle filtering on the online name disambiguation performance using most recent 2 years' records as test set. The results are averaged out over all 25 name references.

tional time depends on other factors, such as the latent dimensionality, the number of particles, and the number of classes generated, which in turn depends on the values of the hyperparameters used in the data model and concentration parameter in dirichlet process.

5.6.7 Study of Parameter Sensitivity

In our proposed Bayesian non-exhaustive classification framework, there are three user-defined parameters, namely latent dimension h, concentration parameter α in Dirichlet process, and number of particles M in particle filtering. In this section, we investigate the classifier performance with respect to these three parameter variations. The results are shown in Figures 5.4(a), 5.4(b), 5.4(c). Specifically, first for latent dimension sensitivity, as we see from Figure 5.4(a), for both online inference approaches, as the latent dimension increases, the online name disambiguation performance in terms of Mean-F1 first increases and then decreases. The possible explanation is that when the latent dimension is too small, the representation capability of the latent feature is not sufficient and we may lose information. However, when the latent dimension is too large, the proposed INNMF technique (details in Section 5.3.1) is too complex and we may over-fit to the data. The second parameter

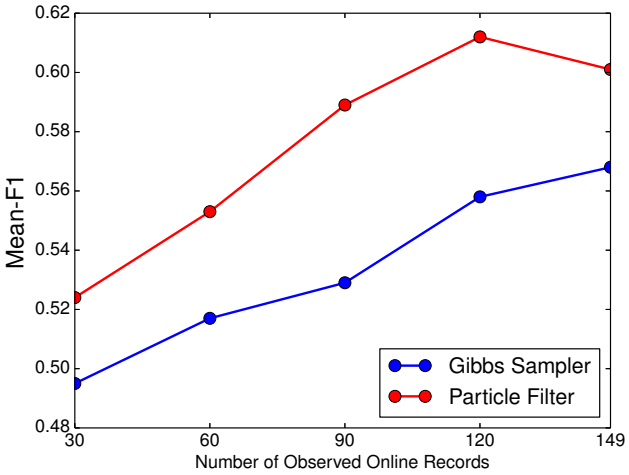


Fig. 5.5.: Online name disambiguation performance over the number of observed online records on name reference "Jing Zhang" using most recent 3 years' records as test set.

α in the Dirichlet process prior model (Section 5.4.1) controls the probability of assigning an incoming record to a new class and it plays a critical role in the number of generated classes in the online name disambiguation process. As we can observe from Figure 5.4(b), the classifier performance is robust with respect to different α values. Finally, Figure 5.4(c) shows that only a few number of particles is sufficient to have desirable classifier prediction (details in Section 5.4.3).

5.6.8 Performance over the Number of Observed Online Records

We investigate the online name disambiguation performance over the number of sequentially observed records. We use the name reference "Jing Zhang" [2] with its corresponding most recent 3 years' records as test set as a case study. Specifically, we evaluate the mean-F1 score as we process $\{30, 60, 90, 120, 149\}$ test records. As we see from Figure 5.5, as more records are observed online, the overall disambiguation

[2]We choose name reference "Jing Zhang" as a case study due to the fact that it contains the largest number of test records (149) among all name references used in the experiment.

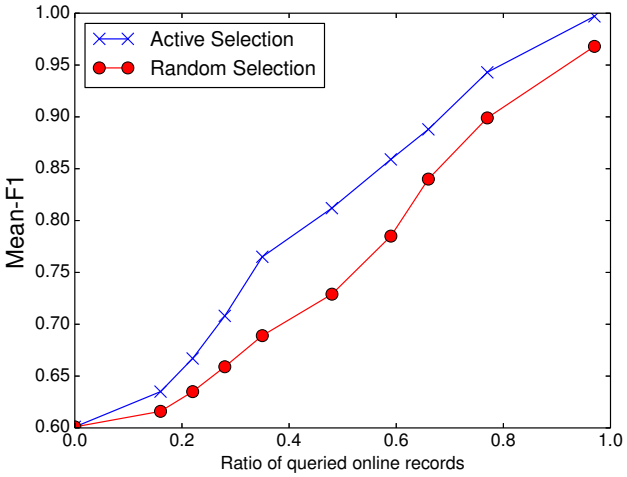


Fig. 5.6.: Mean-F1 comparison between our proposed active selection and random selection with respect to different ratios of queried online records on name reference "Jing Zhang" using most recent 3 years' records as test set. The higher the curve, the better the performance.

performance improves in both one-pass Gibbs sampler and particle filtering based online inference techniques. The results demonstrate that our proposed learning model has self-adjusting capacity that accurately classify incoming online records to both novel and existing classes and effectively prevent error propagation during the online execution stage.

5.6.9 Results of Active Online Name Disambiguation

Now we compare the results of our proposed particle filtering algorithm for online name disambiguation with active selection and random selection (randomly selecting a number of sequentially observed records to query the users for ground-truth). Specifically, for the user-defined interactiveness threshold parameter τ (defined in Section 5.5), we set it to $\{0.1, 0.2, ..., 0.9, 1, 0\}$, and run our proposed active online name disambiguation method 20 times under each τ. Then we compute the average ratio of queried online records for different values of τ. Note that the larger the value

of τ, the fewer the number of queried online records. For the random selection, it queries an online record with probability $0 < p < 1$. In other words, for each online record, the method draws a value from a uniform distribution $U(0, 1)$. If the value is smaller than p, it queries the label. Otherwise, it does not. For a fair comparison, we set p as the ratio of queried online records in our proposed interactive framework. The result on name reference "Jing Zhang" is shown in Figure 5.6.

As we observe, for both active and random selection frameworks, compared to the no feedback scenario where τ is set to be 1.0 and we do not query any online records, incorporating user feedback helps to improve online name disambiguation performance in terms of mean-F1 score. However, our proposed active selection framework is better than random selection consistently under different ratios of queried online records for performing active online name disambiguation. In particular, our proposed active online name disambiguation framework actively queries those records whose label information are uncertain. In contrast, the labels acquired by the random selection may be redundant and lead to the waste of labeling effort. Similar results are obtained for other name references as well.

5.7 Chapter Summary

To conclude, in this chapter we present a Bayesian non-exhaustive classification framework for the task of online name disambiguation. Given sequentially observed online records, our proposed method classifies the incoming records into existing classes, as well as emerging classes by learning posterior probability of a Dirichlet process Gaussian mixture model. Our experimental results on bibliographic datasets demonstrate that the proposed method significantly outperforms the existing state-of-the-arts. As a real-life application, we propose an active online name disambiguation method to improve the prediction accuracy by exploiting user feedback.

6. FEATURE SELECTION FOR CLASSIFICATION UNDER ANONYMITY CONSTRAINT

6.1 Objective and Motivation

In this chapter, inspired by the research on privacy-preserving data publishing [96], we propose a novel privacy metric, called k-anonymity by containment, to measure the potential disclosure risk of textual features used in name disambiguation application. In particular, we propose two privacy-aware feature selection methods to select a subset of textual features such that on the reduced feature set, the data has small disclosure risk, at the same time, the utility of data is maximally preserved for performing name disambiguation. Our proposed method is generic, besides name disambiguation task, we also show two extra real-life applications, namely adult income classification, and spam email filtering.

6.2 Introduction

Over the last decade, with the proliferation of various online platforms, such as web search, eCommerce, social networking, micro-messaging, streaming entertainment and cloud storage, the digital footprint of today's Internet user has grown at an unprecedented rate. At the same time, the availability of sophisticated computing paradigm and advanced machine learning algorithms have enabled the platform owners to mine and analyze tera-bytes of digital footprint data for building various predictive analytics and personalization products. For example, search engines and social network platforms use search keywords for providing sponsored advertisements that are personalized for a user's information need; e-commerce platforms use visitor's search history for bolstering their merchandising effort; streaming entertainment

providers use people's rating data for building future product or service recommendation. However, the impressive personalization of services of various online platforms enlighten us as much, as they do make us feel insecure, which stems from knowing the fact that individual's online behaviors are stored within these companies, and an individual, more often, is not aware of the specific information about themselves that is being stored.

The key reason for a web user's insecurity over the digital footprint data (also known as microdata) is that such data contain sensitive information. For instance, a person's online search about a disease medication may insinuate that she may be suffering from that disease; a fact that she would rather not disclose. Similarly, people's choice of movies, their recent purchases, etc. reveal enormous information regarding their background, preference and lifestyle. Arguably microdata exclude biographical information, but due to the sheer size of our digital footprint the identity of a person can still be recovered from these data by cross-correlation with other data sources or by using publicly available background information. In this sense, these apparently non-sensitive attributes can serve as a quasi-identifier. For an example, Narayanan et al. [114] have identified a Netflix subscriber from his anonymous movie rating by using Internet Movie Database (IMDB) as the source of background information. For the case of Netflix, anonymous microdata was released publicly for facilitating Netflix prize competition, however even if the data is not released, there is always a concern that people's digital footprint data can be abused within the company by employees or by external hackers, who have malicious intents.

For the case of microdata, the identity disclosure risk is high due to some key properties of such a dataset—high-dimensionality and sparsity. Sparsity stands for the fact that for a given record there is rarely any record that is similar to the given record considering full multi-dimensional space. It is also shown that in high-dimensional data, the ratio of distance to the nearest neighbor and the farthest neighbor is almost one, i.e., all the points are far from each other [115]. Due to this fact, privacy is difficult to achieve on such datasets. A widely used privacy metric that quantifies

the disclosure risk of a given data instance is k-anonymity [81], which requires that for any data instance in a dataset, there are at least $k-1$ distinct data instances sharing the same feature vector—thus ensuring that unwanted personal information cannot be disclosed merely through the feature vector. However, for high dimensional data, k-anonymization is difficult to achieve even for a reasonable value of k (say 5); typically, value based generalization or attribute based generalization is applied so that k-anonymity is achieved, but Aggrawal has proved both theoretically and experimentally that for high dimensional data k-anonymity is not a viable solution even for a k value of 2 [115]. He has also shown that as data dimensionality increases, entire discriminatory information in the data is lost during the process of k-anonymization, which severely limits the data utility. Evidently, finding a good balance between a user's privacy and the utility of high dimensional microdata is an unsolved problem—which is the primary focus of this chapter.

A key observation of a real-life high dimensional dataset is that it exhibits high clustering tendency in many sub-spaces of the data, even though over the full dimension the dataset is very sparse. Thus an alternative technique for protecting identity disclosure on such data can be finding a subset of features, such that when projecting on these set of features an acceptable level of anonymity can be achieved. One can view this as column suppression instead of more commonly used row suppression for achieving k-anonymity [81]. Now for the case of feature selection for a given k, there may exist many sub-spaces for which a given dataset satisfies k-anonymity, but our objective is to obtain a set of features such that projecting on this set offers the maximum utility of the dataset in terms of a supervised classification task.

Consider the toy dataset that is shown in Table 6.1. Each row represents a person, and each column (except the first and the last) represents a keyword. If a cell entry is '1' then the keyword at the corresponding column is associated with the person at the corresponding row. Reader may think this table as a tabular representation of the search log of an eCommerce platform, where the '1' under e_i column stands for the fact that the corresponding user has searched using the keyword x_j within a given

Table 6.1.: A toy 2-class dataset with binary feature-set

User	x_1	x_2	x_3	x_4	x_5	Class
e_1	1	0	1	0	1	$+1$
e_2	1	0	1	0	1	-1
e_3	1	0	0	1	1	$+1$
e_4	1	0	1	0	1	$+1$
e_5	1	1	1	0	1	-1
e_6	1	1	0	1	1	-1

Table 6.2.: Projections of the dataset in table 6.1 on two feature-sets (Feature Set-1 and Feature Set-2)

User	Feature Set-1			Class	Feature Set-2		
	x_1	x_2	x_5		x_3	x_4	x_5
e_1	1	0	1	$+1$	1	0	1
e_2	1	0	1	-1	1	0	1
e_3	1	0	1	$+1$	0	1	1
e_4	1	0	1	$+1$	1	0	1
e_5	1	1	1	-1	1	0	1
e_6	1	1	1	-1	0	1	1

period of time, and '0' represents otherwise. The last column represents whether this user has made a purchase over the same time period. The platform owner wants to solve a classification problem to predict which of the users are more likely to make a purchase.

Say, the platform owner wants to protect the identity of its site visitor by making the dataset k-anonymous. Now, for this toy dataset, if he chooses $k = 2$, this dataset is not k-anonymous. For instance, the feature vector of e_3, 10011 is unique in this dataset. However, the dataset is k-anonymous for the same k under the subspace spanned by $\{x_3, x_4, x_5\}$. It is also k-anonymous (again for the same k) under the subspace spanned by $\{x_1, x_2, x_5\}$ (See Table 6.2). Among these two choices, the latter subspace is probably a better choice, as the features in this set are better discriminator than the features in the former set with respect to the class-label. For feature set $\{x_1, x_2, x_5\}$, if we associate the value '101' with the +1 label, and the value '111' with the -1 label, we make only 1 mistake out of 6. On the other hand for feature set $\{x_3, x_4, x_5\}$, no good correlation exists between the feature values and the class labels.

The research problem in the above task is the selection of optimal binary feature set for utility preserving entity anonymization, where the utility is considered with respect to the classification performance and the privacy is guaranteed by enforcing a k-anonymity like constraint [72]. In existing works, k-anonymity is achieved by suppression or generalization of cell values, whereas in this work we consider to achieve the same by selecting an optimal subset of features that maximizes the classification utility of the dataset. Note that, maximizing the utility of the dataset is the main objective of this task, privacy is simply a constraint which a user enforces by setting the value of a privacy parameter based on the application domain and the user's judgment. For the privacy model, we define k-anonymity by containment in short, k-AC (definition forthcoming), where k is the user-defined privacy parameter, which has similar meaning as it has in traditional k-anonymity.

Our choice of k-anonymity like metric over more theoretical counterparts, such as, differential privacy (DP) is due to the pragmatic reason that all existing privacy laws and regulations, such as, HIPAA (Health Information Portability and Accountability Act) and PHIPA (Personal Health Information Protection Act) use k-anonymity. Also, k-anonymity is flexible and simple, thus enabling people to understand and apply it for almost any real-life privacy preserving needs; on the contrary, DP based methods use a privacy parameter (ϵ), which has no obvious interpretation and even by the admission of original author of DP, choosing an appropriate value for this parameter is difficult [86]. Moreover, differential privacy based methods add noise to the data entities, but the decision makers in many application domains (such as, health care), where privacy is an important issue, are quite uncomfortable to the idea of noise imputation [116]. Finally, authors in [117] state that differential privacy is not suitable for protecting large sparse tables produced by statistics agencies and sampling organizations—this disqualifies differential privacy as a privacy model for protecting sparse and very high dimensional user's microdata from the e-commerce and Internet search engines.

6.2.1 Our Contributions

In this work, we consider the task of feature selection under privacy constraint. This is a challenging task, as it is well-known that privacy is always at odds with the utility of a knowledge-based system, and finding the right balance is a difficult task [84, 118]. Besides, feature selection itself, without considering the privacy constraint, is an $\mathcal{NP}$-Hard problem [119].

Given a classification dataset with binary features and an integer k, our proposed solutions find a subset of features such that after projecting each instance on these subsets each entity in the dataset satisfies a privacy constraint, called k-anonymous by containment (k-AC). Our proposed privacy constraint k-AC is an adapted version of k-anonymity, which strikes the correct balance between disclosure risk and dataset

utility, and it is particularly suitable for high dimensional binary data. We also propose two algorithms: `Maximal` and `Greedy`. The first is a maximal itemset mining based method and the second is a greedy incremental approach, both respecting the user-defined AC constraints.

The algorithms that we propose are particularly intended for high dimensional sparse microdata where the features are binary. The nature of such data is different from a typical dataset that is considered in many of the existing works on privacy preserving data disclosure mechanism. The first difference is that existing works consider two kinds of attributes, sensitive and nonsensitive, whereas for our dataset all attributes are considered to be sensitive, and any subset of published attributes can be used by an attacker to de-anonymize one or more entities in the dataset using probabilistic inference methodologies. On the other hand, the unselected attributes are not published so they cannot be used by an attacker to de-anonymize an entity. Second, we only consider binary attributes, which enable us to provide efficient algorithms and an interesting anonymization model. Considering only binary attributes may sound an undue restriction, but in reality binary attributes are adequate (and often preferred) when modeling online behavior of a person, such as 'like' in Facebook, 'bought' in Amazon, and 'click' in Google advertisement. Also, collecting explicit user feedback in terms of frequency data (say, the number of times a search keyword is used) may be costly in many online platforms. Nevertheless, as shown in [114], binary attributes are sufficient for an attacker to de-anonymize a person using high dimensional microdata, so safeguarding user privacy before disclosing such dataset is important.

The contributions of our work are outlined below:

1. We propose a novel method for entity anonymization using feature selection over a set of binary attributes from a two-class classification dataset. For this, we design a new anonymization model, named *k-anonymity by containment (k-AC)*, which is particularly suitable for high-dimensional binary microdata.

2. We propose two methods for solving the above task and show experimental results to validate the effectiveness of these methods.

3. We show the utility of the proposed methods with two real-life applications. Specifically, we show how the privacy-aware feature selection affects their performance.

6.3 Privacy Basics

Given a dataset D, where each row corresponds to a person, and each column contains non-public information about that person; examples include disease, medication, sexual orientation, etc. In the context of online behavior, the search keywords, or purchase history of a person may be such information. Privacy preserving data publishing methodologies make it difficult for an attacker to re-identify a person who is in the dataset. For re-identification, an attacker generally uses a set of attributes that act almost like a key and it uniquely identifies some individual in the datasets. These attributes are called *quasi-identifiers*. *k-anonymity* is a well-known privacy metric defined as below.

Definition 6.3.1 (k-anonymity) A dataset D satisfies k-anonymity if for any row entity $e \in D$ there exist at least $k-1$ other entities that have the same values as e for every possible quasi-identifiers.

The database in Table 6.1 is not 2-anonymous, as the row entity e_3 is unique considering the entire attribute-set $\{x_1, x_2, x_3, x_4, x_5\}$ as quasi-identifier. On the other hand, It is 2-anonymous for both the datasets (one with Feature Set-1 and the other with Feature Set-2) in Table 6.2. For numerical or categorical attributes, a process, called *generalization* and/or suppression (row or cell value) are used for achieving k-anonymity. Generalization partitions the values of an attribute into disjoint buckets and identifies each bucket with a value. Suppression either hides the entire row or some of its cell values, so that the anonymity of that entity can be maintained.

Generalization and suppression make anonymous group, where all the entities in that group have the same value for every possible quasi-identifier, and for a dataset to be k-anonymous, the size of each of such groups is at least k. In this work we consider binary attributes; each such attribute has only two values, 0 and 1. For binary attributes, value based generalization relegates to the process of column suppression, which incurs a loss of data utility. In fact, any form of generalization based k-anonymization incurs a loss in data utility due to the decrement of data variance or due to the loss of discernibility. Suppression of a row is also a loss as in this case the entire row entity is not discernible for any of the remaining entities in the dataset. Unfortunately, most of existing methods for achieving k-anonymity using both generalization and suppression operations do not consider an utility measure targeting supervised classification task.

There are some security attacks against which k-anonymity is vulnerable. For example, k-anonymity is susceptible to both homogeneity and background knowledge based attacks. More importantly, k-anonymity does not provide statistical guaranty about anonymity which can be obtained by using ϵ-differential privacy [86]—a method which provides strong privacy guarantees independent of an adversary's background knowledge. There are existing methods that adopt differential privacy principle for data publishing. Authors in [120,121] propose Laplace mechanism to publish the contingency tables by adding noise generated from a Laplace distribution. However, such methods suffer from the utility loss due to the large amount of added noise during the sanitization process. To resolve this issue, [122] proposes to utilize exponential mechanism for maximizing the trade-off between differential privacy and data utility. However, the selection of utility function used in the exponential mechanism based approach strongly affects the data utility in subsequent data analysis task. In this work, we compare the performance of our proposed privacy model, namely k-AC, with both Laplace and exponential based differential privacy frameworks (See Section 6.6.2) to show that k-AC better preserves the data utility than the differential privacy based methods.

A few works [85, 123] exist which consider classification utility together with k-anonymity based privacy model, but none of them consider feature selection which is the main focus of this work. In one of the earliest works, Iyengar [85] solves k-anonymization through generalization and suppression while minimizing a proposed utility metric called CM (Classification Metric) using a genetic algorithm which provides no optimality guaranty. The CM is defined as below:

Definition 6.3.2 (Classification Metric [85]) Classification metric (CM) is a utility metric for classification dataset, which assigns a penalty of 1 for each suppressed entity, and for each non-suppressed entity it assigns a penalty of 1 if those entities belong to the minority class within its anonymous group. CM value is equal to the sum of penalties over all the entities.

In this work, we compare the performance of our work with CM based privacy-aware utillty metric.

6.4 Problem Statement

Given a classification dataset with binary attributes, our objective is to find a subset of attributes which increase the non-disclosure protection of the row entities, and at the same time maintain the classification utility of the dataset without suppressing any of the row entities. In this section we will provide a formal definition of the problem.

We define a database $D(E, I)$ as a binary relation between a set of entities (E) and a set of features (I); thus, $D \subseteq E \times I$, where $E = \{e_1, e_2, \cdots, e_n\}$ and $I = \{x_1, x_2, \cdots, x_d\}$; n and d are the number of entities and the number of features, respectively. The database D can also be represented as a $n \times d$ binary data matrix, where the rows correspond to the entities, and the columns correspond to the features. For an entity $e_i \in E$, and a feature $x_j \in I$, if $\langle e_i, x_j \rangle \in D$, the corresponding data matrix entry $D(e_i, x_j) = 1$, otherwise $D(e_i, x_j) = 0$. Thus each row of D is a binary vector of size d in which the 1 entries correspond to the set of features with which

the corresponding row entity is associated. In a classification dataset, besides the attributes, the entities are also associated to a class label which is a category value. In this task we assume a binary class label $\{C_1, C_2\}$. A typical supervised learning task is to use the features I to predict the class label of an entity.

We say that an entity $e_i \in E$ contains a set of features $X = \{x_{i1}, x_{i2}, \cdots, x_{il}\}$, if $D(e_i, x_{ik}) = 1$ for all $k = 1, 2, \cdots, l$; set X is also called *containment set* of the entity e_i.

Definition 6.4.1 (Containment Set) Given a binary dataset, $D(E, I)$, the containment set of a row entity $e \in E$, represented as $CS_D(e)$, is the set of attributes $X \subseteq I$ such that $\forall x \in X, D(e, x) = 1$, and $\forall y \in I - X$, $D(e, y) = 0$.

When the dataset D is clear from the context we will simply write $CS(e)$ instead of $CS_D(e)$ to represent the containment set of e.

Definition 6.4.2 (k-anonymity by containment) In a binary dataset $D(E, I)$ and for a given positive integer k, an entity $e \in E$ satisfies k-anonymity by containment if there exists a set of entities $F \subseteq E$, such that $e \notin F \wedge |F| \geq k - 1 \wedge \forall f \in F, CS_D(f) \supseteq CS_D(e)$. In other words, their exist at least $k - 1$ other entities in D such that their containment set is the same or a superset of $CS_D(e)$.

By definition, if an entity satisfies k-anonymity by containment, it satisfies the same for all integer values from 1 upto k. We use the term $AC(e)$ to denote the largest k for which the entity e satisfies k-anonymity by containment.

Definition 6.4.3 (k-anonymous by Containment Group) For a binary dataset $D(E, I)$, if $e \in E$ satisfies the k-anonymity by containment, k-anonymous by containment group with respect to e exists and this is $F \cup \{e\}$, where F is the largest possible set as is defined in Definition 6.4.2.

Definition 6.4.4 (k-anonymous by Containment Dataset) A binary dataset $D(E, I)$ is k-anonymous by containment if every entity $e \in E$ satisfies k-anonymity by containment.

We extend the term AC over a dataset as well, thus $AC(D)$ is the smallest k for which the dataset D is anonymous.

Example: For the dataset in Table 6.1, $CS(e_1) = \{x_1, x_3, x_5\}$. Entity e_1 satisfies 4-anonymity by containment, because for each of the following three entities e_2, e_4, and e_5, their containment sets are the same or supersets of $CS(e_1)$. But, the entity e_6 only satisfies 1-anonymity by containment, as besides itself no other entity contains $CS(e_6) = \{x_1, x_2, x_4, x_5\}$. 4-anonymous by containment group of e_2 exists, and it is $\{e_1, e_2, e_4, e_5\}$, but 5-anonymous by selection group for the same entity does not exist. The dataset in Table 6.1 is 1-anonymous by containment because there exists one entity, namely e_6 such that the highest k-value for which e_6 satisfies k-anonymity by containment is 1; alternatively $AC(D) = 1$

k-anonymity by containment (k-AC) is the privacy metric that we use in this work. The argument for this metric is that if a large number of other entities contain the same or super feature subset which an entity e contains, the disclosure protection of the entity e is strong, and vice versa. Thus a higher value of k stands for a higher level of privacy for e. k-anonymity by containment (k-AC) is similar to k-anonymity for binary feature set except that for k-AC only the '1' value of feature set is considered as a privacy risk. It is easy to see that k-anonymity by containment (k-AC) is a relaxation of k-anonymity. In fact, the following lemma holds.

Lemma 1 *If a dataset satisfies k-anonymity for a k value, it also satisfies k-AC for the same k-value, but the reverse does not hold necessarily.*

Proof Say, the dataset D satisfies k-anonymity; then for any row entity $e \in D$, there exists at least $k-1$ other row entities with identical row vector as e. Containment set of all these $k-1$ entities is identical to e, so e satisfies k-AC. Since this holds for all $e \in D$, the dataset D satisfies k-AC.

To prove that the reverse does not hold, we will give a counter-example. Assume D has three entities and two features with the following feature values, $D = \{(1,1),(1,0),(1,1)\}$. D satisfies 2-AC because the smallest anonymity by contain-

ment value of the entities in the dataset is 2. But, the dataset does not satisfy 2-anonymity because the entity $(1, 0)$ is unique in the dataset. ■

However, the relaxed privacy that k-AC provides is adequate for disclosure protection in a high dimensional sparse microdata with binary attributes, because k-AC conceals the list of the attributes in a containment set of an entity, which could reveal sensitive information about the entity. For example, if the dataset is about the search keywords that a set of users have used over a given time, for a person having a 1 value under a keyword potentially reveals sensitive information about the behavior or preference of that person. Having a value of 0 for a collection of features merely reveals the knowledge that the entity is *not associated* with that attribute. In the online microdata domain, due to the high dimensionality of the data, non-association with a set of attributes is not a potential privacy risk. Also note that, in traditional datasets, only a few attributes which belong to non-sensitive group are assumed to be quasi-identifier, so a privacy metric, like k-anonymity works well for such dataset. But, for high-dimensional dataset, k-anonymity is severely restrictive and utility loss of data by column suppression is substantial because feature subsets containing very small number of features pass k-anonymity criteria. On the other hand, k-AC based privacy metric enables selection of sufficient number of features for retaining the classification utility of the dataset. In short, k-AC retains the classification utility substantially, whereas k-anonymity fails to do so for most high dimensional data.

Feature selection [119] for a classification task is to select a subset of highly predictive variables so that classification accuracy possibly improves which happens due to the fact that contradictory or noisy attributes are generally ignored during the feature selection step. For a dataset $D(E, I)$, and a feature-set $S \subseteq I$, following relational algebra notations, we use $\Pi_S(D)$ to denote the projection of database D over the feature set S. Now, given a user-defined integer number k, our goal is to perform an optimal feature selection on the dataset D to obtain $\Pi_S(D)$ which satisfies two objectives: first, $\Pi_S(D)$ is k-anonymous by containment, i.e., $AC(\Pi_S(D)) \geq k$; second, $\Pi_S(D)$ maintains the predictive performance of the classification task as much

as possible. Selecting a subset of features is similar to the task of column suppression based privacy protection, but the challenge in our task is that we want to suppress columns that risk privacy, and at the same time we want to retain columns that have good predictive performance for a downstream supervised classification task using the sanitized dataset. For denoting the predictive performance of a dataset (or a projected dataset) we define a classification utility function f. The higher the value of f, the better the dataset for the classification. We consider f to be a filter based feature selection criteria which is independent of the classification model that we use.

The formal research task of this work is as below. Given a binary dataset $D(E, I)$, and an integer number k, find $S \subseteq I$ so that $f(\Pi_S(D))$ is maximized under the constraint that $AC(\Pi_S(D)) \geq k$. Mathematically,

$$\begin{aligned} \underset{S \subseteq I}{\text{maximize}} \quad & f(\Pi_S(D)) \\ \text{subject to} \quad & AC(\Pi_S(D)) \geq k \end{aligned} \tag{6.1}$$

Due to the fact that the problem 6.1 is a combinatorial optimization problem (optimizing over the space of feature subsets) which is NP-Hard, here we propose two effective local optimal solutions for this problem.

6.5 Methods

In this section, we describe two algorithms, namely `Maximal` and `Greedy` that we propose for the task of feature selection under privacy constraint. `Maximal` is a maximal itemset mining based feature selection method, and `Greedy` is a greedy method with privacy constraint based filtering. In the following subsections, we discuss them in detail.

6.5.1 Maximal Itemset Based Approach

A key observation regarding k-anonymity by containment (AC) of a dataset is that this criteria satisfies the downward-closure property under feature selection. The following lemma holds:

Lemma 2 *Say $D(E, I)$ is a binary dataset and $X \subseteq I$ and $Y \subseteq I$ are two feature subsets. If $X \subseteq Y$, then $AC(\Pi_X(D)) \geq AC(\Pi_Y(D))$.*

PROOF*: Let's prove by contradiction. Suppose $X \subseteq Y$ and $AC(\Pi_X(D)) < AC(\Pi_Y(D))$. Then from the definition of AC, there exists at least one entity $e \in E$ for which $AC(\Pi_X(e)) < AC(\Pi_Y(e))$. Now, let's assume A_X and A_Y are the set of entities which make the anonymous by containment group for the entity e in $\Pi_X(D)$ and $\Pi_Y(D)$, respectively. Since $AC(\Pi_X(e)) < AC(\Pi_Y(e)), |A_X| < |A_Y|$; so there exists an entity $p \in A_Y \setminus A_X$, for which $CS_{\Pi_Y(D)}(p) \supseteq CS_{\Pi_Y(D)}(e)$ and $CS_{\Pi_X(D)}(p) \not\supseteq CS_{\Pi_X(D)}(e)$; But this is impossible, because $X \subseteq Y$, if $CS_{\Pi_Y(D)}(p) \supseteq CS_{\Pi_Y(D)}(e)$ holds, then $CS_{\Pi_X(D)}(p) \supseteq CS_{\Pi_X D}(e)$ must be true. Thus, the lemma is proved by contradiction.*

Let's call the collection of feature subsets that satisfy the AC threshold for a given k the feasible set, and represent it with $\mathcal{F}_k$. Thus, $\mathcal{F}_k = \{X \mid X \subseteq I \wedge AC(\Pi_X(D)) \geq k\}$. A subset of features $X \in \mathcal{F}_k$ is called maximal if it has no supersets that are feasible. Let $\mathcal{M}_k$ be the set of all maximal subset of features. Then $\mathcal{M}_k = \{X \mid X \in \mathcal{F}_k \wedge \nexists Y \supset X, \text{such that } Y \in \mathcal{F}_k\}$. As we can observe given an integer k, if there exists a maximal feature set Z that satisfies the AC constraint, then any feature set $X \subseteq Z$, also satisfies the same AC constraint, i.e., $k \leq AC(\Pi_Z(D)) \leq AC(\Pi_X(D))$ if $X \subseteq Z \in \mathcal{M}_k$ based on the Lemma 2.

Example: For the dataset in Table 6.1, the 2-anonymous by containment feasible feature set [1] $\mathcal{F}_2 = \{x_1, x_2, x_3, x_4, x_5, x_1x_2, x_1x_3, x_1x_4, x_1x_5, x_2x_5, x_3x_5, x_4x_5, x_1x_2x_5, x_1x_3x_5, x_1x_4x_5\}$ and $\mathcal{M}_2 = \{x_1x_2x_5, x_1x_3x_5, x_1x_4x_5\}$. In this dataset, the feature-set $x_2x_3 \notin \mathcal{F}_2$ because in $\Pi_{x_2x_3}(D)$, $CS(e_5) = \{x_2, x_3\}$ and the size of the k-anonymous

[1] To enhance the readability, we write the feature set as string; for example, the set $\{x_1, x_2\}$ is written as x_1x_2.

by containment group of e_5 is 1; thus $AC(\Pi_{x_2x_3}(D)) = 1 < 2$. On the other hand for feature-set $x_1x_2x_5$, the projected dataset $\Pi_{x_1x_2x_5}(D)$ has two k-anonymous by containment groups, which are $\{e_1, e_2, e_3, e_4, e_5, e_6\}$ and $\{e_5, e_6\}$; since each group contains at least two entities, $AC(\Pi_{x_1x_2x_5}(D)) = 2$

Lemma 3 *Say, $D(E, I)$ is a binary dataset, and T is its transaction representation where each entity $e \in E$ is a transaction consisting of the containment set $CS_D(e)$. Frequent itemset of the dataset T with minimum support threshold k are the feasible feature set $\mathcal{F}_k$ for the optimization problem 6.1.*

PROOF*: Say, X is a frequent itemset in the transaction T for support threshold k. Then, the support-set of X in T are the transactions (or entities) which contain X. Since, X is frequent, the support-set of X consists of at least k entities. In the projected dataset $\Pi_X(D)$, all these entities make a k-anonymous by containment group, thus satisfying k-anonymity by containment. For each of the remaining entities (say, e), e's containment set contains some subset of X (say Y) in $\Pi_X(D)$. Since, X is a frequent itemset and $Y \subset X$, Y is also frequent with a support-set that has at least k entities. Then e also belongs to a k-anonymous by containment group. Thus, each of the entities in D belongs to some k-anonymous by containment group(s) which yields: X is frequent $\Rightarrow X \in \mathcal{F}_k$. Hence proved.*

A consequence of Lemma 2 is that for a given dataset D, an integer k, and a feature set S, if $AC(\Pi_S(D)) \geq k$, any subset of S (say, R) satisfies $AC(\Pi_R(D)) \geq k$. This is identical to the downward closure property of frequent itemset mining. Also, Lemma 3 confirms that any itemset that is frequent in the transaction representation of D for a minimum support threshold k is a feasible solution for problem 6.1. Hence, an apriori like algorithm for itemset mining can be used for effectively enumerating all the feature subsets of D which satisfies the required k-anonymity by containment constraint.

6.5.1.1 Maximal Feasible Feature Set Generation

For large datasets, the feasible feature set $\mathcal{F}_k$ which consists of feasible solutions for the optimization problem 6.1 can be very large. One way to control its size is by choosing appropriate k; if k increases, $|\mathcal{F}_k|$ decreases, and vice-versa, but choosing a large k negatively impacts the classification utility of the dataset, thus reducing the optimal value of problem (6.1). A brute force method for finding the optimal feature set S is to enumerate all the feature subset in $\mathcal{F}$ and find the one that is the best given the utility criteria f. However, this can be very slow. So, `Maximal` generates all possible maximal feature sets $\mathcal{M}_k$ instead of generating $\mathcal{F}_k$ and search for the best feature subset within $\mathcal{M}_k$. The idea of enumerating $\mathcal{M}_k$ instead of $\mathcal{F}_k$ comes from the assumption that with more features the classification performance will increase; thus, the size of the feature set is its utility function value; i.e., $f(\Pi_S(D)) = |S|$, and in that case the largest set in $\mathcal{M}_k$ is the solution to the problem 6.1.

An obvious advantage of working only with the maximal feature set is that for many datasets, $|\mathcal{M}_k| << |\mathcal{F}_k|$, thus finding solution within $\mathcal{M}_k$ instead of $\mathcal{F}_k$ leads to significant savings in computation time. Just like the case for frequent itemset mining, maximal frequent itemset mining algorithm can also be used for finding $\mathcal{M}_k$. Any off-the-shelf software can be used for this. In `Maximal`algorithm we use the LCM-Miner package provided in[2] which, at present, is the fastest method for finding maximal frequent itemsets.

6.5.1.2 Classification Utility Function

The simple utility function $f(\Pi_S(D)) = |S|$ has a few limitations. First, the ties are very commonplace, as there are many maximal feature sets that have the same size. Second, and more importantly, this function does not take into account the class labels of the instances so it cannot find a feature set that maximizes the separation between the positive and negative instances. So, we consider another utility function,

[2] `http://research.nii.ac.jp/~uno/code/lcm.html`

named as $HamDist$, which does not succumb as much to the tie situation. It also considers the class label for choosing features that provide good separation between the positive and negative classes.

Definition 6.5.1 (Hamming Distance) For a given binary database $D(E, I)$, and a subset of features, $S \subseteq I$, the Hamming distance between $\Pi_S(a)$ and $\Pi_S(b)$ is defined as below:

$$d_H(\Pi_S(a), \Pi_S(b)) = \sum_{j=1}^{|S|} \mathbb{1}\{a_{s_j} \neq b_{s_j}, \forall s_j \in S\} \tag{6.2}$$

where $\mathbb{1}\{X\}$ is the indicator function, and a_{s_j} and b_{s_j} are the s_jth feature value under S for the entities a and b, respectively.

We can partition the entities in $D(E, I)$ into two disjoint subsets, E_1 and E_2; entities in E_1 have a class label value of C_1, and entities in E_2 have a class label value of C_2.

Definition 6.5.2 (*HamDist*) Given a dataset $D(E = E_1 \cup E_2, I)$ where the partitions E_1 and E_2 are based on class labels, the classification utility function $HamDist$ for a feature subset $S \subseteq I$ is the average Hamming distance between all pair of entities a and b such that $a \in E_1$ and $b \in E_2$.

$$HamDist(S) = \frac{1}{\mid E_1 \mid\mid E_2 \mid} \sum_{a \in E_1, b \in E_2} d_H(\Pi_S(a), \Pi_S(b)) \tag{6.3}$$

Example: For the dataset in Table 6.1, for its projection on $x_1x_2x_5$ (see, Table 2), distance of e_1 from the negative entities are $0+1+1 = 2$, and the same for the other positive entities, e_3 and e_4 also. So, $HamDist(x_1x_2x_5) = ((0+1+1)+(0+1+1)+(0+1+1))/9 = 6/9$. From the same table we can also see that $HamDist(x_3x_4x_5) = 8/9$.

As we can observe from Equations 6.2 and 6.3, the utility function $HamDist(S)$ reflects the discriminative power between classes given the feature set S. The larger the value of $HamDist(S)$, the better the quality of selected feature set S to distinguish between classes. Another separation metric similar to $HamDist$ is *DistCnt* (Distinguish Count), which is defined below.

Definition 6.5.3 (*DistCnt)* For $D(E_1 \cup E_2, I)$, and $S \subseteq I$, $DistCnt$ is the number of pairs from E_1 and E_2 which can be distinguished using at least one feature in S. Mathematically,

$$DistCnt(S) = \frac{1}{\mid E_1 \mid\mid E_2 \mid} \sum_{a \in E_1, b \in E_2} \mathbb{1}\{\Pi_S(a) \neq \Pi_S(b)\} \tag{6.4}$$

$DistCnt$ can also be used instead of $HamDist$ in the `Maximal` algorithm. Note that, we can also use CM criterion (see Definition 6.3.2) instead of $HamDist$; however, experimental results show that CM performs much poorer in terms of AUC. Besides, both $HamDist$ and $DistCnt$ functions have some good properties (will be discussed in Section 6.5.2) which CM does not have.

The `Maximal` algorithm utilizes classification utility metrics ($HamDist$ or $DistCnt$) for selecting the best feature set from the maximal set $\mathcal{M}_k$. For some datasets, the size of $\mathcal{M}_k$ can be large and selecting the best feature set by applying the utility metric on each element of $\mathcal{M}_k$ can be time-consuming. Then, we can find the best feature set among the largest sized element in $\mathcal{M}_k$. Another option is to consider the maximal feature sets in $\mathcal{M}_k$ in the decreasing order of their size in such a way that at most r of the maximal feature sets from set $\mathcal{M}_k$ are chosen as candidates for which the utility metric computation is performed. In this work we use this second option by setting $r = 20$ for all our experiments.

6.5.1.3 Maximal Itemset Based Method (Pseudo-code)

The pseudo-code for `Maximal` is given in Algorithms 5. `Maximal` takes privacy parameter k and the number of maximal patterns r as input and returns the final feature set S which satisfies k-anonymity by containment. Line 1 uses the LCM-Miner to generate all the maximal feature sets that satisfy k-anonymity by containment for the given k value. Line 2 groups maximal feasible feature sets according to its size and selects top r maximal feature sets with the largest size and builds the candidate feature sets. Then the algorithm computes the feature selection criteria $HamDist$ of

each feature set in the candidate feature sets and returns the best feature set that has the maximum value for this criteria.

Algorithm 5 Maximal Itemset Mining Based Feature Selection Method

Input: $D(E, I)$, k, r
Output: S
1: Calculate maximal feature set $\mathcal{M}_k$ which contains the feature-sets satisfying k-AC for the given k
2: Select best feature-set S based on the $HamDist$ criteria by considering r largest-sized feature set in $\mathcal{M}_k$.
3: **return** S

The complexity of the above algorithm predominantly depends on the complexity of the maximal itemset mining step (Line 1), which depends on the input value k. For larger k, the privacy is stronger and it reduces $\mathcal{M}_k$ making the algorithm run faster, but the classification utility of the dataset may suffer. On the other hand, for smaller k, $\mathcal{M}_k$ can be large making the algorithm slower, but it better retains the classification utility of the dataset.

6.5.2 Greedy with Modular and Sub-Modular Objective Functions

A potential limitation of `Maximal` is that for dense datasets this method can be slow. So, we propose a second method, called `Greedy` which runs much faster as it greedily adds a new feature to an existing feasible feature-set. For greedy criteria, `Greedy` can use different separation functions which discriminate between positive and negative instances. In this work we use $HamDist$ (See Definition 6.5.2) and $DistCnt$ (See Definition 6.5.3). Thus `Greedy` solves the Problem (6.1) by replacing f by either of the two functions. Because of the monotone property of these functions, `Greedy` ensures that as we add more features, the objective function value of (6.1) monotonically increases. The process stops once no more features are available to add to the existing feature set while ensuring the desired AC value of the projected dataset.

6.5.2.1 Submodularity, and Modularity

Definition 6.5.4 (Submodular Set Function) Given a finite ground set U, a monotone function f that maps subsets of U to a real number $f : 2^U \rightarrow \mathbb{R}$ is called submodular if

$$f(S \cup \{u\}) - f(S) \geq f(T \cup \{u\}) - f(T), \forall S \subseteq T \subseteq U, u \in U$$

If the above condition is satisfied with equality, the function is called modular.

Theorem 6.5.1 *HamDist is monotone, submodular, and modular.*

Proof For a dataset $D(E, I)$, $S \subseteq I$, and $T \subseteq I$ are two arbitrary feature-sets, such that $S \subseteq T$. $E = E_1 \cup E_2$, where the partition is based on class label. Consider the pair (a, b), such that $a \in E_1$ and $b \in E_2$. Let, $w(\cdot)$ be a function that sums the Hamming distance over all such pairs (a, b) for a given feature subset S. Thus, $w(S) = \sum_{a \in E_1, b \in E_2} d_H(\Pi_S(a), \Pi_S(b))$, where the function d_H is the Hamming distance between a and b as defined in Equation 6.2. Similarly we can define $w(T)$, for the feature subset T. Using Equation 6.2, $d_H(\Pi_S(a), \Pi_S(b))$ is the summation over each of the features in S. Since $S \subseteq T$, $d_H(\Pi_T(a), \Pi_T(b))$ includes the sum values for the variables in S and possibly includes the sum value of other variables, which is non-negative. Summing over all (a, b) pairs yields $w(S) \leq w(T)$. So, *HamDist* is monotone. Now, for a feature $u \notin T$,

$$\begin{aligned}
w(S \cup \{u\}) &= \sum_{a \in E_1, b \in E_2} d_H\left(\Pi_{S \cup \{u\}}(a), \Pi_{S \cup \{u\}}(b)\right) \\
&= \sum_{a \in E_1, b \in E_2} \sum_{s_j \in S \cup \{u\}} \mathbb{1}\{a_{s_j} \neq b_{s_j}\} \text{(using Eq. 6.2)} \\
&= \sum_{a \in E_1, b \in E_2} \left(\sum_{s_j \in S} \mathbb{1}\{a_{s_j} \neq b_{s_j}\} + \mathbb{1}\{a_u \neq b_u\} \right) \\
&= w(S) + w(\{u\})
\end{aligned}$$

Similarly, $w(T \cup \{u\}) = w(T) + w(\{u\})$. Then, we have $w(\{u\}) = w(S \cup \{u\}) - w(S) = w(T \cup \{u\}) - w(T)$. Dividing both sides by $1/(|E_1| \cdot |E_2|)$ yields $HamDist(S \cup \{u\}) - HamDist(S) = HamDist(T \cup \{u\}) - HamDist(T)$. Hence proved with the equality. ■

Theorem 6.5.2 *DistCnt is monotone, and submodular.*

Proof Given a dataset $D(E, I)$ where E is partitioned as $E_1 \cup E_2$ based on class label. Now, consider a bipartite graph, where vertices in one partition (say, V_1) correspond to features in I, and the vertices of other partition (say, V_2) correspond to a distinct pair of entities (a, b) such that $a \in E_1$, and $b \in E_2$; thus, $|V_2| = |E_1| \cdot |E_2|$. If for a feature $x \in V_1$, we have $a_x \neq b_x$, an edge exists between the corresponding vertices $x \in V_1$ and $(a, b) \in V_2$. Say, $S \subseteq V_1$ and $T \subseteq V_1$ and $S \subseteq T$. For a set of vertices, $\Gamma(\cdot)$ represents their neighbor-list. Since, the size of neighbor-list of a vertex-set is monotone and submodular, for $u \notin T$, we have $|\Gamma(S)| \leq |\Gamma(T)|$, and $|\Gamma(S \cup \{u\})| - |\Gamma(S)| \geq |\Gamma(T \cup \{u\})| - |\Gamma(T)|$. By construction, for a feature set, S, $\Gamma(S)$ contains the entity-pairs for which at least one feature-value out of S is different. Thus, *DistCnt* function is $\frac{|\Gamma(\cdot)|}{|V_2|}$ and it is submodular. ■

Algorithm 6 Greedy Algorithm for $HamDist$

Input: $D(E, I)$, k
Output: S

```
1: Sort the features in non-increasing order based on their hamDist, denoted as F_sorted
2: S = ∅
3: for each feature x ∈ F_sorted do
4:    if AC(Π_{S∪{x}}(D)) ≥ k then
5:       S = S ∪ {x}
6:    else
7:       break
8:    end if
9: end for
10: return S
```

Theorem 6.5.3 *For monotone submodular function f, let S be a set of size k obtained by selecting elements one at a time, each time choosing an element provides the largest marginal increase in the function value. Let S^* be a set that maximizes the value of f over all k-element sets. Then $f(S) \geq (1 - \frac{1}{e})f(S^*)$; in other words, S provides $(1 - \frac{1}{e})$-approximation. For modular function $f(S) = f(S^*)$ [124].*

Algorithm 7 Greedy Algorithm for *DistCnt*

Input: $D(E, I)$, k
Output: S

```
1: T = ∅
2: repeat
3:    S = T
4:    ΔH_max = 0.0
5:    for u ∈ I \ S do
6:       Compute ΔH = DistCnt(S ∪ {u}) − DistCnt(S)
7:       if ΔH > ΔH_max then
8:          ΔH_max = ΔH
9:          u_m = u
10:      end if
11:   end for
12:   T = S ∪ {u_m}
13: until AC(Π_T(D)) ≥ k
14: return S
```

6.5.2.2 Greedy Method (Pseudo-code)

Using the above theorems we can design two greedy algorithms, one for modular function *HamDist*, and the other for submodular function *DistCnt*. The pseudo-codes of these algorithms are shown in Algorithm 6 and Algorithm 7. Both the methods take binary dataset D and integer value k as input and generate the selected feature set S as output. Initially $S = \emptyset$. For modular function, the marginal gain of an added feature can be pre-computed, so Algorithm 6 first sorts the features in non-increasing order of their *HamDist* values, and greedily adds features until it encounters a feature such that its addition does not satisfy the *AC* constraint. For

Table 6.3.: Statistics of real-world datasets

Dataset	# Entities	# Features	# Pos	# Neg	Density
Name Disambiguation	148	552	74	74	9.7%
Email	1099	24604	618	481	0.9%

submodular function *DistCnt*, margin gain cannot be pre-computed, so Algorithm 7 selects the new feature by iterating over all the features and finding the best one (Line 5 -11). The terminating condition of this method is identical to Algorithm 6. Since the number of features is finite, both the methods always terminate with a valid S which satisfies $AC(\Pi_S(D)) \geq k$.

Compared to `Maximal`, both greedy methods are faster. With respect to number of features (d), Algorithm 6 runs in $O(d \lg d)$ time and Algorithm 7 runs in $O(d^2)$ time. Also, using Theorem 6.5.3, Algorithm 2 returns the optimal size $|S|$ feature-set, and Algorithm 3 returns S, for which the objective function value is $(1 - 1/e)$ optimal over all possible size-$|S|$ feature sets.

6.6 Experiments and Results

In order to evaluate our proposed methods we perform various experiments. Our main objective in these experiments is to validate how the performance of the proposed privacy preserving classification varies as we change the value of AC—user-defined privacy threshold metric. We also compare the performance of our proposed utility preserving anonymization methods with other existing anonymization methods such as k-anonymity and differential privacy. It is important to note that we do not claim that our methods provide a better utility with identical privacy protection as other methods, rather we claim that our methods provide adequate privacy protection which is suitable for high dimensional sparse microdata with a much superior AUC value—a classification utility metric which we want to maximize in our problem setup. We use two real-world datasets for our experiments. Both datasets consist of

entities that are labeled with 2 classes. The number of entities, the number of features, the distribution of the two classes (#postive and #negative), and the dataset density (fraction of non-zero cell values) are shown in Table 6.3.

6.6.1 Privacy Preserving Classification Tasks

Below, we discuss the datasets and the privacy preserving classification tasks that we solve using our proposed methods.

Name Disambiguation (ND) [12]. The objective of this classification task is to identify whether the name reference at a row in the data matrix maps to multiple real-life persons or not. Such an exercise is quite common in the Homeland Security for disambiguating multiple suspects from their digital footprints [11, 37]. Privacy of the people in such a dataset is important as many innocent persons can also be listed as a suspect. Given a set of keywords that are associated with a name reference, we build a binary data matrix for solving the ND task. We use Arnetminer academic publication data. In this dataset, each row is a name reference of one or multiple researchers, and each column is a research keyword within the computer science research umbrella. A '1' entry represents that the name reference in the corresponding row has used the keyword in her (or their) published works. In our dataset, there are 148 rows which are labeled such that half of the people in this dataset are pure entity (a negative case), and the rest of them are multi-entity (a positive case). The dataset contains 552 attributes (keywords).

To solve the name disambiguation problem we first perform topic modeling over the keywords and then compute the distribution of entity u's keywords across different topics. Our hypothesis is that for a pure entity the topic distribution will be concentrated on a few related topics, but for an impure entity (which is composed of multiple real-life persons) the topic distribution will be distributed over many non-related top-

ics. We use this idea to build a simple classifier which uses an entropy-based score $E(u)$ for an entity u as below:

$$E(u) = -\sum_{k=1}^{|T|} P(u \mid T_k) \log P(u \mid T_k) \tag{6.5}$$

where $P(u \mid T_k)$ is the probability of u belonging to topic T_k, and $|T|$ represents the pre-defined number of topics for topic modeling. Clearly, for a pure entity the entropy-based score $E(u)$ is relatively smaller than the same for a non-pure entity. We use this score as our predicted value and compute AUC (area under ROC curve) to report the performance of the classifier.

Email The last dataset, namely Email dataset [3] is a collection of approximately 1099 personal email messages distributed in 10 different directories. Each directory contains both legitimate and spam messages. To respect the privacy issue, each token including word, number, and punctuation symbol is encrypted by a unique number. The classification task is to distinguish the spam email with nonspam email. We use this data to mimic microdata (such as twitter or Facebook messages) classification. Privacy is important in such a dataset as keyword based features in a micro-message can potentially identify a person. In the dataset, each row is an email message, and each column denotes a token. A '1' in a cell represents that the row reference contains the token in the email message.

6.6.2 Experimental Setting

For our experiments, we vary the k value of the proposed k-anonymity by containment (AC) metric and run `Maximal` and different variants of `Greedy` independently for building projected classification datasets for which AC value is at least k. We use the names $HamDist$ and $DistCnt$ for the two variants of `Greedy` (Algorithm 6 and 7), which optimize Hamming distance and Distinguish count greedy criteria, respectively.

[3] `http://www.csmining.org/index.php/pu1-and-pu123a-datasets.html`

As we mentioned earlier, k-anonymity based method imposes strong restriction which severely affects the utility of the dataset. To demonstrate that, instead of using AC, we utilize k-anonymity as our privacy criteria for different variants of `Greedy`. We call these competing methods k-anonymity $HamDist$, and k-anonymity $DistCnt$. It is important to note that, in our experiments under the same k setting, the k-anonymity based competing methods may not provide the same level of privacy. For instance, for the same k value, privacy protection of our proposed method $HamDist$ is not the same as that of the k-anonymity $HamDist$, simply because k-AC is a relaxation of k-anonymity.

We also use four other methods for comparing their performance with the performance of our proposed methods. We call these competing methods RF [125], CM Greedy [85], Laplace-DP, and Exponential-DP. We discuss these methods below.

RF is a Randomization Flipping based k-anonymization technique presented in [125], which randomly flips the feature value such that each instance in the dataset satisfies the k-anonymity privacy constraint. RF uses clustering such that after random flipping operation, each cluster has at least k entities with the same feature values with respect to the entire feature set.

CM greedy represents another greedy based method which uses Classification Metric utility criterion proposed in [85] as utility metric (See definition 6.3.2). It assigns a generalization penalty over the rows of the dataset and uses a genetic algorithm for the classification task, but for a fair comparison we use CM criterion in the `Greedy` algorithm and with the selected features we use identical setup for classification.

Laplace-DP [18] is a method to use feature selection for ϵ-differential private data publishing. Authors in [18] utilize Laplace mechanism [121] for ϵ-differential privacy guarantee. To compare with their method, we first compute the utility of each feature $x_i \in I$ as its true output using $HamDist$ function in Definition 6.5.2 denoted as $H(x_i)$. Then we add independently generated noise according to a Laplace distribution with $Lap(\frac{\Delta H}{\epsilon})$ to each of the $|I|$ outputs, and the noisy output for each feature x_i is defined as $\hat{H(x_i)} = H(x_i) + Lap(\frac{\Delta H}{\epsilon})$, where ΔH is the sensitivity of $HamDist$ function. After

that we select top-N features by considering N largest noisy outputs. On the reduced dataset, we apply a private data release method which provides ϵ-differential privacy guaranty. The general philosophy of this method is to first derive a frequency matrix of the reduced dataset over the feature domain and add Laplace noise with $Lap(\frac{1}{\epsilon})$ to each count (known as marginal) to satisfy the ϵ-differential privacy. Then the method adds additional data instances to match the above count. Such an approach is discussed in [86] as a private data release mechanism.

Exponential-DP is another ϵ-differential privacy aware feature selection method. Compared to the work presented in [18], we use exponential mechanism [122] based ϵ-differential privacy to select features. In particular, we choose each feature $x_i \in I$ with probability proportional to $exp(\frac{\epsilon}{2\Delta H}H(x_i))$. That is, the feature with a higher utility score in terms of $HamDist$ function is exponentially more likely to be chosen. The private data release stage of Exponential-DP is as same as the one in Laplace-DP. Note that, for both Laplace-DP and Exponential-DP, prior feature selection is essential for such methods to reduce the data dimensionality, otherwise the number of marginals is an intractable number ($2^{|\mathcal{F}|}$, for a binary dataset with $\mathcal{F}$ features) and adding instances to match count for each such instance is practically impossible.

For all the algorithms and all the datasets (except ND) we use the LibSVM to perform SVM classification using L2 loss with 5-fold cross validation. The only parameter for libSVM is regularization-loss trade-off C which we tune using a small validation set. For each of the algorithms, we report AUC and the selected feature count (SFC). For RF method, it selects all the features, so for this method we report the percentage of cell values for which the bit is flipped. We use different k-anonymity by containment (AC) values in our experiments. For practical k-anonymization, k value between 5 and 10 is suggested in the earlier work [72]; we use three different k values, which are $5, 8$ and 11. For a fair comparison, for both Laplace and Exponential DP, we use the same number of features as is obtained for the case of $HamDist$ Greedy under $k = 5$. Since k-anonymity and differential privacy use totally different parameter setting mechanisms (one based on k, and the other based on ϵ), it is not

Table 6.4.: AUC comparison among different privacy methods for the name disambiguation task

Method	AUC (Selected Feature Count)		
	k=5	k=8	k=11
`Maximal`	0.82 (61)	0.81 (43)	0.79 (32)
HamDist	**0.88 (27)**	**0.88 (24)**	**0.81 (16)**
DistCnt	0.81 (11)	0.81 (11)	0.80 (10)
CM Greedy [85]	0.68 (2)	0.68 (2)	0.68 (2)
RF [125]	0.75±0.02 (11.99%)	0.73±0.03 (14.03%)	0.72±0.02 (16.49%)
k-anonymity *HamDist*	0.55 (3)	0.55 (2)	0.55 (2)
k-anonymity *DistCnt*	0.79 (3)	0.79 (3)	0.77 (2)
Full-Feature-Set		0.87 (552)	

easy to understand what value of ϵ in DP will make a fair comparison for a k value of 5 in k-AC. So, for both Laplace-DP and exponential-DP, we show the differential privacy results for different ϵ values: $0.5, 1.0, 1.5$, and 2.0 . Note that the original work [86] has suggested to use a value of 1.0 for ϵ. While using DP based methods, we distribute half of the privacy budget for the feature selection step and the remaining half to add noise into marginals in the private data release step. Moreover, in the feature selection procedure, we further equally divide the budget for the selection of each feature.

RF, Laplace-DP, and Exponential-DP are randomized methods, so for each dataset we run all of them 10 times and report the average AUC and standard deviation. For each result table in the following sections, we also highlight the best results in terms of AUC among all methods under same k setting. We run all the experiments on a 2.1 GHz Machine with 4GB memory running Linux operating system.

6.6.3 Name Disambiguation

In Table 6.4 we report the AUC value of anonymized name disambiguation task using various privacy methods (in rows) for different k values (in columns). For better comparison, our proposed methods, competing methods, and non-private methods

are grouped by the horizontal lines: our proposed methods are in the top group, the competing methods are in the middle group, and non-private methods are in the bottom group. For differential privacy comparison, we show the AUC result in Figure 6.1(a) 6.1(b). For each method, we also report the count of selected features (SFC). Since RF method uses the full set of features; for this method the value in the parenthesis is the percent of cell values that have been flipped. We also report the AUC performance using full feature set (last row). As non-private method in bottom group has no privacy restriction, thus the result is independent of k.

For most of the methods increasing k decreases the number of selected features, which translates to poorer classification performance; this validates the privacy-utility trade-off. However, for a given k, our proposed methods perform better than the competing methods in terms of AUC metric for all different k values. For instance, for $k = 5$, the AUC result of RF and CM Greedy are only 0.75 and 0.68 respectively, whereas different versions of proposed `Greedy` obtain AUC values between 0.81 and 0.88. Among the competing methods, both Laplace-DP and Exponential-DP perform the worst (0.51 AUC under $\epsilon = 1.0$) as shown in the first group of bars in Figure 6.1(a) & 6.1(b), and k-anonymity $DistCnt$ performs the best (0.79 for k=5); yet all completing methods perform much poorer than our proposed methods. A reason for this may be most of the competing methods are too restrictive, as we can see that they are able to select only 2 to 3 features for various k values. In comparison, our proposed methods are able to select between 11 and 61 features, which help our methods to retain classification utility. The bad performance of differential privacy based methods is due to the fact that in such a setting, the added noise is too large in both feature selection and private data release steps. In general, the smaller the ϵ, the stronger privacy guarantee the differential privacy provides. However, stronger privacy protection in terms of ϵ always leads to worse data utility in terms of AUC as shown in Figure 6.1(a) 6.1(b). Therefore, even though differential privacy provides stronger privacy guarantee, the utility of data targeting supervised classification task is significantly destroyed. For this dataset, we observe that the performance of RF

Table 6.5.: Comparison among different privacy methods for Email dataset using AUC

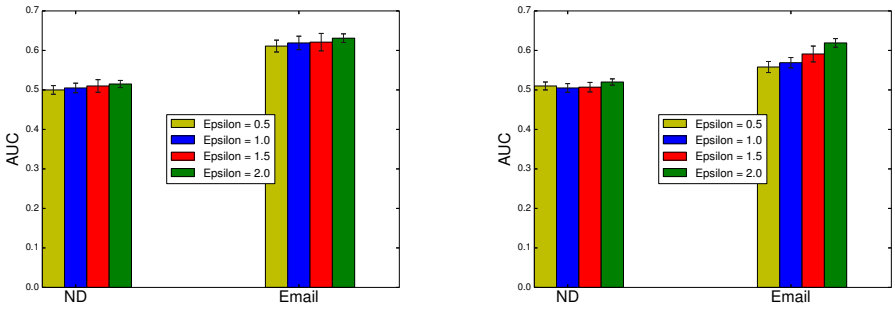
Method	AUC (Selected Feature Count)		
	k=5	k=8	k=11
`Maximal`	0.94 (121)	0.92 (66)	0.90 (58)
HamDist	0.91 (11)	0.91 (11)	0.91 (11)
DistCnt	**0.95 (11)**	**0.93 (7)**	**0.93 (7)**
CM Greedy [85]	0.86 (3)	0.86 (3)	0.86 (3)
RF [125]	0.87±0.02 (1.30%)	0.86±0.01 (1.73%)	0.87±0.03 (2.03%)
k-anonymity *HamDist*	0.84 (4)	0.84 (4)	0.84 (4)
k-anonymity *DistCnt*	0.81 (4)	0.81 (4)	0.81 (4)
Full-Feature-Set	0.95 (24604)		

is largely dependent on the percentage of flips in the cell-value; if this percentage is large, the performance is poor. As k increases, with more privacy requirement, the percentage of flips increases, and the AUC drops.

For a sparse dataset like the one that we use for name disambiguation, feature selection helps classification performance. In this dataset, using full set of features (no privacy), we obtain only 0.87 AUC value, whereas using less than 10% of features we can achieve comparable or better AUC using our proposed methods (when k=5). Even for $k = 11$, our methods retain substantial part of the classification utility of the dataset and obtain AUC value of 0.81 (see second row). Also, note that under $k = 5$ and 8, our *HamDist* performs better than using full feature set, which demonstrates our proposed privacy-aware feature selection methods not only have the competitive AUC performance, but provide strong privacy guarantees as well.

6.6.4 Spam Email Filtering

In Table 6.5, we compare AUC value of different methods for spam email filtering task. This is very high dimensional data with 24604 features. As we can observe, our proposed methods, especially *DistCnt* and *HamDist* perform better than the competing methods. For example, for $k = 5$, the classification AUC of RF is 0.87 with



(a) Laplace mechanism based differential privacy

(b) Exponential mechanism based differential privacy

Fig. 6.1.: Classification performance of differential privacy based methods for different ϵ values on three datasets. Laplace mechanism is on the left and exponential mechanism is on the right. Each group of bars belong to one specific dataset, and within a group different bars represent different ϵ values.

flip rate 1.30%, but using less than 0.045% of features *DistCnt* obtains an AUC value of 0.95, which is equal to the AUC value using the full feature set. Again, k-anonymity based methods show worse performance as they select less number of features due to stronger restriction of this privacy metric. For instance, for $k = 5$, $HamDist$ selects 11 features, but k-anonymity $HamDist$ selects only 4 features. Due to this, classification results using k-anonymity constraint are worse compared to those using our proposed AC as privacy metric. As shown in Figure 6.1(a) 6.1(b), both Laplace-DP and Exponential-DP with various privacy budget ϵ setups perform much worse than all the competing methods in Table 6.5, which demonstrates that the significant amount of added noise during the sanitization process deteriorates the data utility and leads to bad classification performance. Among our methods, both *DistCnt* and `Maximal` are the best as they consistently hold the classification performance for all different k settings.

6.7 Chapter Summary

In this chapter, we propose a novel method for entity anonymization using feature selection. We define a new anonymity metric called k-anonymity by containment which is particularly suitable for high dimensional microdata. We also propose two feature selection methods along with two classification utility metrics. These metrics satisfy submodular properties, thus they enable effective greedy algorithms. In experiment section we show that both proposed methods select good quality features on a variety of datasets for retaining the classification utility yet they satisfy the user defined anonymity constraint.

7. FUTURE WORK AND CONCLUSION

Name disambiguation using relational, streaming, and privacy-preserving textual data are novel research problems, and our works are just the beginning, thus the opportunity for the future works is abundant. In this chapter, we discuss a few possible future directions to explore.

For the works presented in Chapter 3 and Chapter 4, the validity of current methodologies are particularly linked to academic collaboration networks. However, the exploration of ground-truth name disambiguation datasets in security sensitive domains, such as email, phone call and online social networks, is an interesting future research direction. Another future work is to improve current network embedding model presented in Chapter 4. For instance, we can incorporate contextual network topological information as a Laplacian regularizer into the optimization framework to alleviate the sparseness of networks for further improving the name disambiguation performance. In addition, we can also utilize neural network models to capture the high-order non-linear relationship between nodes in the graph.

For the active online name disambiguation presented in Chapter 5, there are rooms to improve the method proposed in this work as well. The data model used in this study is limited with the Gaussian distribution. The proposed approach can be extended to problems involving more flexible class distributions by choosing a mixture model for each class data and a hierarchical Dirichlet Process Prior model over class distributions. Another future work would be to use time-dependent Dirichlet process to incorporate temporal information into the prior model. Specifically, for bibliographic data, a clear temporal trend exists; most people in academia start with $1-2$ papers per year, and then increase this rate significantly during their career's peaks which diminish as they get closer to retirement. Incorporating such intuition into the model may also improve the online name disambiguation performance substantially.

For name disambiguation using privacy-preserving textual data presented in Chapter 6, we consider binary and categorical features. An immediate future work is to extend this work on datasets with real-valued features. Another one would be to consider the sensitivity level of features. Finally, it will be practical to consider absent attributes in the proposed k-anonymity by containment privacy metric.

To conclude, in this dissertation, we introduce several practical settings for the task of name disambiguation. Specifically, we formulate the name disambiguation using relational data in the form of anonymized graphs, and designed effective disambiguation-aware features to solve the problem. We also developed Bayesian non-exhaustive classification framework for the task of active online name disambiguation by considering the streaming nature of data records. Finally, we designed a novel privacy metric, called k-anonymity by containment, to measure the potential disclosure risk of textual features used in the name disambiguation application.

REFERENCES

REFERENCES

[1] V. T. Chakaravarthy, H. Gupta, P. Roy, and M. K. Mohania, "Efficient techniques for document sanitization," in *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, 2008, pp. 843–852.

[2] R. Bunescu and M. Pasca, "Using encyclopedic knowledge for named entity disambiguation," in *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*, 2006, pp. 9–16.

[3] L. Cen, E. C. Dragut, L. Si, and M. Ouzzani, "Author disambiguation by hierarchical agglomerative clustering with adaptive stopping criterion," in *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2013, pp. 741–744.

[4] S. Zwicklbauer, C. Seifert, and M. Granitzer, "Robust and collective entity disambiguation through semantic embeddings," in *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2016, pp. 425–434.

[5] X. Han, L. Sun, and J. Zhao, "Collective entity linking in web text: A graph-based method," in *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2011, pp. 765–774.

[6] J. Hoffart, M. A. Yosef, I. Bordino, H. Fürstenau, M. Pinkal, M. Spaniol, B. Taneva, S. Thater, and G. Weikum, "Robust disambiguation of named entities in text," in *Proceedings of the 21st Conference on Empirical Methods in Natural Language Processing*, 2011, pp. 782–792.

[7] X. Han and J. Zhao, "Named entity disambiguation by leveraging wikipedia semantic knowledge," in *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, 2009, pp. 215–224.

[8] J. Tang, A. C. M. Fong, B. Wang, and J. Zhang, "A unified probabilistic framework for name disambiguation in digital library," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 6, pp. 975–987, 2012.

[9] X. Wang, J. Tang, H. Cheng, and P. S. Yu, "Adana: Active name disambiguation," in *IEEE 11th International Conference on Data Mining*, 2011, pp. 794–803.

[10] H. Han, L. Giles, H. Zha, C. Li, and K. Tsioutsiouliklis, "Two supervised learning approaches for name disambiguation in author citations," in *Proceedings of the 4th ACM/IEEE Joint Conference on Digital Libraries*, 2004, pp. 296–305.

[11] B. Zhang, T. K. Saha, and M. Al Hasan, "Name disambiguation from link data in a collaboration graph," in *IEEE/ACM 5th International Conference on Advances in Social Networks Analysis and Mining*, pp. 81–84.

[12] T. K. Saha, B. Zhang, and M. Al Hasan, "Name disambiguation from link data in a collaboration graph using temporal and topological features," *Social Network Analysis Mining*, vol. 5, no. 1, pp. 11:1–11:14, 2015.

[13] B. Zhang and M. A. Hasan, "Name disambiguation in anonymized graphs using network embedding," in *Proceedings of the 26th ACM International on Conference on Information and Knowledge Management*, 2017.

[14] B. Zhang, M. Dundar, and M. A. Hasan, "Bayesian non-exhaustive classification a case study: Online name disambiguation using temporal record streams," in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM, 2016, pp. 1341–1350.

[15] ——, "Bayesian non-exhaustive classification for active online name disambiguation," *arXiv preprint arXiv:1702.02287*, 2017.

[16] J. Vaidya and C. Clifton, "Privacy-preserving k-means clustering over vertically partitioned data," in *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2003, pp. 206–215.

[17] J. Vaidya, M. Kantarc, and C. Clifton, "Privacy-preserving naive Bayes classification," *Very Large Data Bases Journal*, vol. 17, no. 4, pp. 879–898, 2008.

[18] Y. Jafer, S. Matwin, and M. Sokolova, "Using feature selection to improve the utility of differentially private data publishing," *Computer Science*, vol. 37, pp. 511–516, 2014.

[19] E. Pattuk, M. Kantarcioglu, H. Ulusoy, and B. Malin, "Privacy-aware dynamic feature selection," in *2015 IEEE 31st International Conference on Data Engineering*, 2015, pp. 78–88.

[20] B. Zhang, N. Mohammed, V. S. Dave, and M. A. Hasan, "Feature selection for classification under anonymity constraint," *Transactions on Data Privacy*, vol. 10, no. 1, pp. 1–25, 2017.

[21] S. Cucerzan, "Large-scale named entity disambiguation based on Wikipedia data," in *Proceedings of the 17th Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 2007, pp. 708–716.

[22] S. S. Kataria, K. S. Kumar, R. R. Rastogi, P. Sen, and S. H. Sengamedu, "Entity disambiguation with hierarchical topic models," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2011, pp. 1037–1045.

[23] H. Han, H. Zha, and C. L. Giles, "Name disambiguation in author citations using a k-way spectral clustering method," in *Proceedings of the 5th ACM/IEEE-CS Joint Conference on Digital Libraries*, 2005, pp. 334–343.

[24] Y. F. Tan, M. Y. Kan, and D. Lee, "Search engine driven author disambiguation," in *Proceedings of the 6th ACM/IEEE-CS Joint Conference on Digital Libraries*, 2006, pp. 314–315.

[25] X. Yin, J. Han, and P. Yu, "Object distinction: Distinguishing objects with identical names," in *Proceedings of the 23rd IEEE International Conference on Data Engineering*, 2007, pp. 1242–1246.

[26] S. Li, G. Cong, and C. Miao, "Author name disambiguation using a new categorical distribution similarity," in *Proceedings of 22nd European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2012, pp. 569–584.

[27] Y. Song, J. Huang, I. G. Councill, J. Li, and C. L. Giles, "Efficient topic-based unsupervised name disambiguation," in *Proceedings of the 7th ACM/IEEE-CS Joint Conference on Digital Libraries*. ACM, 2007, pp. 342–351.

[28] D. Zhang, J. Tang, J. Li, and K. Wang, "A constraint-based probabilistic framework for name disambiguation," in *Proceedings of the 16th ACM Conference on Conference on Information and Knowledge Management*, 2007, pp. 1019–1022.

[29] E. Minkov, W. W. Cohen, and A. Y. Ng, "Contextual search and name disambiguation in email using graphs," in *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2006, pp. 27–34.

[30] P. Sen, "Collective context-aware topic models for entity disambiguation," in *Proceedings of the 21st International Conference on World Wide Web*, 2012, pp. 729–738.

[31] Y. Li, C. Wang, F. Han, J. Han, D. Roth, and X. Yan, "Mining evidences for named entity disambiguation," in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2013, pp. 1070–1078.

[32] Y. Cao, J. Li, X. Guo, S. Bai, H. Ji, and J. Tang, "Name list only? target entity disambiguation in short texts." in *Proceedings of the 20th Conference on Empirical Methods in Natural Language Processing*, 2015, pp. 654–664.

[33] B. Malin, "Unsupervised name disambiguation via social network similarity," in *SDM'05 Workshop on Link Analysis, Counterterrorism, and Security*, pp. 93–102.

[34] W.-S. Chin and Y.-C. Juan, "Effective string processing and matching for author disambiguation," in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining Cup Workshop*, 2013, pp. 7:1–7:9.

[35] J. Liu, K. H. Lei, J. Y. Liu, C. Wang, and J. Han, "Ranking-based name matching for author disambiguation in bibliographic data," in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining Cup Workshop*, 2013, pp. 8:1–8:8.

[36] C. Sun, D. Shen, Y. Kou, T. Nie, and G. Yu, "Topological features based entity disambiguation," *Journal of Computer Science and Technology*, vol. 31, no. 5, pp. 1053–1068, 2016.

[37] L. Hermansson, T. Kerola, F. Johansson, V. Jethava, and D. Dubhashi, "Entity disambiguation in anonymized graphs using graph kernels," in *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management*, 2013, pp. 1037–1046.

[38] Y. Qian, Q. Zheng, T. Sakai, J. Ye, and J. Liu, "Dynamic author name disambiguation for growing digital libraries," *Journal of Information Retrieval*, vol. 18, no. 5, pp. 379–412, 2015.

[39] M. Khabsa, P. Treeratpituk, and C. L. Giles, "Online person name disambiguation with constraints," in *Proceedings of the 15th ACM/IEEE-CS Joint Conference on Digital Libraries*. ACM, 2015, pp. 37–46.

[40] A. Veloso, A. A. Ferreira, M. A. Goncalves, A. H. F. Laender, and W. M. Jr., "Cost-effective on-demand associative author name disambiguation." *Information Process Management*, 2012.

[41] A. P. de Carvalho, A. A. Ferreira, A. H. F. Laender, and M. A. Goncalves, "Incremental unsupervised name disambiguation in cleaned digital libraries," *Journal of Information and Data Management*, vol. 2, no. 3, pp. 289–304, 2011.

[42] A. Davis, A. Veloso, A. S. da Silva, W. Meira, Jr., and A. H. F. Laender, "Named entity disambiguation in streaming data," in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, 2012, pp. 815–824.

[43] J. Hoffart, Y. Altun, and G. Weikum, "Discovering emerging entities with ambiguous names," in *Proceedings of the 23rd International Conference on World Wide Web*. ACM, 2014, pp. 385–396.

[44] J. Huang, S. Ertekin, and C. L. Giles, "Efficient name disambiguation for large-scale databases," in *Proceedings of European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2006, pp. 536–544.

[45] Y. Cheng, Z. Chen, J. Wang, A. Agrawal, and A. Choudhary, "Bootstrapping active name disambiguation with crowdsourcing," in *Proceedings of the 22nd ACM International on Conference on Information and Knowledge Management*, 2013, pp. 1213–1216.

[46] T. A. Godoi, R. d. S. Torres, A. M. Carvalho, M. A. Gonçalves, A. A. Ferreira, W. Fan, and E. A. Fox, "A relevance feedback approach for the author name disambiguation problem," in *Proceedings of the 13th ACM/IEEE-CS Joint Conference on Digital Libraries*, 2013, pp. 209–218.

[47] A. A. Ferreira, M. A. Goncalves, and A. H. Laender, "A brief survey of automatic methods for author name disambiguation," *ACM Special Interest Group on Management of Data Record*, vol. 41, no. 2, pp. 15–26, 2012.

[48] S. Chang, W. Han, J. Tang, G.-J. Qi, C. C. Aggarwal, and T. S. Huang, "Heterogeneous network embedding via deep architectures," in *Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015, pp. 119–128.

[49] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2014, pp. 701–710.

[50] J. Tang, M. Qu, and Q. Mei, "Pte: Predictive text embedding through large-scale heterogeneous text networks," in *Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015, pp. 1165–1174.

[51] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proceedings of the 24th International Conference on World Wide Web*, 2015, pp. 1067–1077.

[52] A. Grover and J. Leskovec, "Node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 855–864.

[53] S. Cao, W. Lu, and Q. Xu, "Grarep: Learning graph representations with global structural information," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, 2015, pp. 891–900.

[54] T. Chen and Y. Sun, "Task-guided and path-augmented heterogeneous network embedding for author identification," in *Proceedings of the 10th ACM International Conference on Web Search and Data Mining*, 2017, pp. 295–304.

[55] S. Wang, J. Tang, C. Aggarwal, and H. Liu, "Linked document embedding for classification," in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, 2016, pp. 115–124.

[56] F. Nie, W. Zhu, and X. Li, "Unsupervised large graph embedding," in *AAAI Conference on Artificial Intelligence*, 2017.

[57] C. Tu, H. Liu, Z. Liu, and M. Sun, "Cane: Context-aware network embedding for relation modeling," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, 2017, pp. 1722–1731.

[58] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 1225–1234.

[59] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang, "Community preserving network embedding," in *Proceedings of the 31st Conference on Artificial Intelligence*, 2017, pp. 203–209.

[60] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Y. Chang, "Network representation learning with rich text information," in *Proceedings of the 24th International Conference on Artificial Intelligence*, 2015, pp. 2111–2117.

[61] Z. Stan, J. Meng, Y. Quan, Q. Bing, L. Ting, and Z. ChengXiang, "Contextcare: Incorporating contextual information networks to representation learning on medical forum data," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 2017, pp. 3497–3503.

[62] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proceedings of the 26th International Conference on Neural Information Processing Systems*, 2013, pp. 3111–3119.

[63] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proceedings of the 31st International Conference on Machine Learning*, 2014, pp. 1188–1196.

[64] B. Shaw and T. Jebara, "Structure preserving embedding," in *Proceedings of the 26th Annual International Conference on Machine Learning*, 2009, pp. 937–944.

[65] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, pp. 2323–2326, 2000.

[66] P. Y. Chen, S. Choudhury, and A. O. Hero, "Multi-centrality graph spectral decompositions and their application to cyber intrusion detection," in *Proceedings of the 34th IEEE International Conference on Acoustics, Speech and Signal Processing*, 2016, pp. 4553–4557.

[67] F. Akova, M. Dundar, V. J. Davisson, E. D. Hirleman, A. K. Bhunia, J. P. Robinson, and B. Rajwa, "A machine-learning approach to detecting unknown bacterial serovars." *Statistical Analysis and Data Mining*, vol. 3, no. 5, pp. 289–301, 2010.

[68] M. Dundar, F. Akova, A. Qi, and B. Rajwa, "Bayesian nonexhaustive learning for online discovery and modeling of emerging classes," in *Proceedings of the 29th International Conference on Machine Learning*, 2012, pp. 113–120.

[69] D. J. Miller and J. Browning, "A mixture model and em-based algorithm for class discovery, robust classification, and outlier rejection in mixed labeled/unlabeled data sets," *IEEE Transactions on Pattern Recognition and Machine Intelligence*, pp. 1468–1483, 2003.

[70] M. M. Masud, J. Gao, L. Khan, J. Han, and B. M. Thuraisingham, "Classification and novel class detection in concept-drifting data streams under time constraints," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 6, pp. 859–874, 2011.

[71] T. Al-Khateeb, M. M. Masud, L. Khan, C. C. Aggarwal, J. Han, and B. M. Thuraisingham, "Stream classification with recurring and novel class detection using class-based ensemble," in *Proceedings of the 12th IEEE International Conference on Data Mining*, 2012, pp. 31–40.

[72] L. Sweeney, "k-anonymity: A model for protecting privacy," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 5, pp. 557–570, Oct. 2002.

[73] N. Li and T. Li, "t-closeness: Privacy beyond k-anonymity and l-diversity," in *Proceedings of the 23rd IEEE International Conference on Data Engineering*, 2007, pp. 106–115.

[74] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam, "l-diversity: Privacy beyond k-anonymity," *ACM Transactions on Knowledge Discovery from Data*, vol. 1, no. 1, Mar. 2007.

[75] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Foundation and Trends in Theoretical Computer Science*, vol. 9, pp. 211–407, Aug. 2014.

[76] S. E. Fienberg and J. Jin, "Privacy-preserving data sharing in high dimensional regression and classification settings," *Journal of Privacy and Confidentiality*, vol. 4, no. 1, p. 10, 2012.

[77] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke, "Privacy preserving mining of association rules," in *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2002, pp. 217–228.

[78] L. Bonomi and L. Xiong, "Mining frequent patterns with differential privacy," *Proceedings of Very Large Data Bases Endowment*, vol. 6, no. 12, pp. 1422–1427, Aug. 2013.

[79] R. Bhaskar, S. Laxman, A. Smith, and A. Thakurta, "Discovering frequent patterns in sensitive data," in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2010, pp. 503–512.

[80] M. Atzori, F. Bonchi, F. Giannotti, and D. Pedreschi, "k-anonymous patterns," in *Proceedings of the 9th European Conference on Principles and Practice of Knowledge Discovery in Databases*, 2005, pp. 10–21.

[81] P. Samarati, "Protecting respondents' identities in microdata release," *IEEE Transactions on Knowledge and Data Engineering*, vol. 13, no. 6, pp. 1010–1027, 2001.

[82] A. Meyerson and R. Williams, "On the complexity of optimal k-anonymity," in *Proceedings of the 23rd ACM Symposium on Principles of Database Systems*, 2004, pp. 223–228.

[83] R. J. Bayardo and R. Agrawal, "Data privacy through optimal k-anonymization," in *Proceedings of the 21st International Conference on Data Engineering*, 2005, pp. 217–228.

[84] D. Kifer and J. Gehrke, "Injecting utility into anonymized datasets," in *Proceedings of the 22nd ACM International Conference on Management of Data*, 2006, pp. 217–228.

[85] V. S. Iyengar, "Transforming data to satisfy privacy constraints," in *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2002, pp. 279–288.

[86] C. Dwork, "Differential privacy: A survey of results," in *Proceedings of the 5th International Conference on Theory and Applications of Models of Computation*, 2008, pp. 1–19.

[87] H. Nguyen, A. Imine, and M. Rusinowitch, "Network structure release under differential privacy," *Transactions on Data Privacy*, vol. 9, no. 3, pp. 215–241, 2016.

[88] J. Lee and C. Clifton, "Differential identifiability," in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2012, pp. 1041–1049.

[89] R. Chen, Q. Xiao, Y. Zhang, and J. Xu, "Differentially private high-dimensional data publication via sampling-based inference," in *Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015, pp. 129–138.

[90] W. Qardaji, W. Yang, and N. Li, "Practical differentially private release of marginal contingency tables," in *Proceedings of the 19th ACM International Conference on Management of Data*, pp. 1435–1446.

[91] J. Soria-Comas, J. Domingo-Ferrer, D. Sanchez, and S. Martinez, "Enhancing data utility in differential privacy via microaggregation-based k-anonymity," in *Very Large Data Bases Journal*, vol. 23, no. 5, 2014, pp. 771–794.

[92] R. Chen, B. C. Desai, N. Mohammed, L. Xiong, and B. C. M. Fung, "Publishing set-valued data via differential privacy," in *Proceedings of the Very Large Data Bases Endowment*, vol. 4, no. 11, 2011, pp. 1087–1098.

[93] N. Mohammed, R. Chen, B. C. Fung, and P. S. Yu, "Differentially private data release for data mining," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2011, pp. 493–501.

[94] Y. Jafer, S. Matwin, and M. Sokolova, "Task oriented privacy preserving data publishing using feature selection," in *Proceedings 27th Canadian Conference on Advances in Artificial Intelligence*, 2014, pp. 143–154.

[95] N. Matatov, L. Rokach, and O. Maimon, "Privacy-preserving data mining: A feature set partitioning approach," *Information Sciences*, vol. 180, no. 14, pp. 2696 – 2720, 2010.

[96] B. Fung, K. Wang, R. Chen, and P. S. Yu, "Privacy-preserving data publishing: A survey of recent developments," *ACM Computing Surveys*, vol. 42, no. 4, p. 14, 2010.

[97] F. Wang, J. Li, J. Tang, J. Zhang, and K. Wang, "Name disambiguation using atomic clusters," in *Proceedings of the 9th International Conference on Web-Age Information Management*, 2008, pp. 357–364.

[98] Bhattacharya and L. Getoor, "A latent dirichlet model for unsupervised entity resolution," in *Proceedings of the 6th SIAM Conference on Data Mining*, 2006.

[99] M. J. Zaki and J. Wagner Meira, *Data Mining and Analysis: Fundamental Concepts and Algorithms*. Cambridge University Press, 2014.

[100] P.-Y. Chen, B. Zhang, M. A. Hasan, and A. O. Hero, "Incremental method for spectral clustering of increasing orders," in *KDD Workshop on Mining and Learning with Graphs*, 2016.

[101] M. E. J. Newman, "Modularity and community structure in networks," *Proceedings of the National Academy of Sciences*, vol. 103, no. 23, pp. 8577–8582, 2006.

[102] P. Allison and J. S. Long, "Interuniversity mobility of academic scientists," *American Sociological Research*, vol. 52, no. 5, pp. 643–652, 1987.

[103] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," in *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, 2009, pp. 452–461.

[104] B. Zhang, S. Choudhury, M. A. Hasan, X. Ning, K. Agarwal, S. Purohit, and P. G. P. Cabrera, "Trust from the past: Bayesian personalized ranking based link prediction in knowledge graphs," in *SDM Workshop on Mining Networks and Graphs*, 2016.

[105] W. Zhang, T. Chen, J. Wang, and Y. Yu, "Optimizing top-n collaborative filtering via dynamic negative item sampling," in *Proceedings of the 36th International ACM Conference on Research and Development in Information Retrieval*, 2013, pp. 785–788.

[106] T. Zhao, J. McAuley, and I. King, "Leveraging social connections to improve personalized ranking for collaborative filtering," in *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, 2014, pp. 261–270.

[107] D. Kuang, H. Park, and C. H. Q. Ding, "Symmetric non-negative matrix factorization for graph clustering," in *Proceedings of 16th SIAM International Conference on Data Mining*, 2012, pp. 106–117.

[108] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Advances in Neural Information Processing Systems*, 2001, pp. 556–562.

[109] W. Y. Teh, *Encyclopedia of Machine Learning*, 2010, ch. Dirichlet Process.

[110] J. Sethuraman, "A constructive definition of dirichlet priors," *Statistica Sinica*, pp. 639–650, 1994.

[111] T. W. Anderson, Ed., *An Introduction to Multivariate Statistical Analysis*, 1984.

[112] A. Doucet, S. Godsill, and C. Andrieu, "On sequential Monte Carlo sampling methods for Bayesian filtering," *Statistics and Computing*, vol. 10, pp. 197–208, 2000.

[113] T. Greene and W. S.Rayens, "Partially pooled covariance matrix estimation in discriminant analysis," *Communications in Statistics*, pp. 3679–3702, 1989.

[114] A. Narayanan and V. Shmatikov, "Robust de-anonymization of large sparse datasets," in *Proceedings of the 16th IEEE Symposium on Security and Privacy*, 2008, pp. 111–125.

[115] C. C. Aggarwal, "On k-anonymity and the curse of dimensionality," in *Proceedings of the 31st International Conference on Very Large Data Bases*, 2005, pp. 901–909.

[116] F. K. Dankar and K. E. Emam, "Practicing differential privacy in health care: A review," in *Transactions on Data Privacy*, vol. 6, no. 1, 2013, pp. 35–67.

[117] S. E. Fienberg, A. Rinaldo, and X. Yang, "Differential privacy and the risk-utility tradeoff for multi-dimensional contingency tables," in *Proceedings of the 27th International Conference on Privacy in Statistical Databases*, 2010, pp. 187–199.

[118] D. Kifer and A. Machanavajjhala, "No free lunch in data privacy," in *Proceedings of the 19th ACM International Conference on Management of Data*, 2011, pp. 193–204.

[119] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, Mar. 2003.

[120] B. Barak, K. Chaudhuri, C. Dwork, S. Kale, F. McSherry, and K. Talwar, "Privacy, accuracy, and consistency too: A holistic solution to contingency table release," in *Proceedings of the 26th ACM Symposium on Principles of Database Systems*, 2007, pp. 273–282.

[121] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Proceedings of the 3rd Conference on Theory of Cryptography*, 2006, pp. 265–284.

[122] F. McSherry and K. Talwar, "Mechanism design via differential privacy," in *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science*, 2007, pp. 94–103.

[123] F. Prasser, R. Bild, J. Eicher, H. Spengler, F. Kohlmayer, and K. A. Kuhn, "Lightning: Utility-driven anonymization of high-dimensional data," *Transactions on Data Privacy*, vol. 9, no. 2, pp. 161–185, 2016.

[124] M. Conforti and G. Cornujols, "Submodular set functions, matroids and the greedy algorithm: Tight worst-case bounds and some generalizations of the rado-edmonds theorem," *Discrete Applied Mathematics*, vol. 7, no. 3, pp. 251 – 274, 1984.

[125] J.-W. Byun, A. Kamra, E. Bertino, and N. Li, "Efficient k-anonymization using clustering techniques," in *Proceedings of the 12th International Conference on Database Systems for Advanced Applications*, 2007, pp. 188–200.

VITA

VITA

Baichuan Zhang received his B.S. in electrical engineering from East China University of Science and Technology (ECUST) in June 2012. After graduation, in August 2013, he attended Purdue University to pursue his PhD degree in computer science under the supervision of Dr. Mohammad Al Hasan. His research focuses on information retrieval, deep learning, recommendation systems, graph analysis and data privacy. During his PhD study, he interned at Pacific Northwest National Laboratory, Hortonworks Inc, and Facebook Inc in summer 2015, 2016, 2017 respectively.