UNIVERSITÄT KOBLENZ · LANDAU
Faculty 4: Computer Science

WeST People and Knowledge Networks
Institute for Web Science and Technologies

Author Name Disambiguation using Markov Chain Monte Carlo (MCMC)

Master's Thesis

in partial fulfillment of the requirements for
the degree of Master of Science (M.Sc.)
in Web and Data Science

submitted by
Nagaraj Bahubali Asundi

First supervisor: Dr. Zeyd Boukhers
Institute for Web Science and Technologies

Second supervisor: Dr. Claudia Schon
Institute for Web Science and Technologies

Koblenz, October 2022

Statement

I hereby certify that this thesis has been composed by me and is based on my own work, that I did not use any further resources than specified – in particular no references unmentioned in the reference section – and that I did not submit this thesis to another examination before. The paper submission is identical to the submitted electronic version.

	Yes	No
I agree to have this thesis published in the library.	☒	☐
I agree to have this thesis published on the Web.	☒	☐
The thesis text is available under a Creative Commons License (CC BY-SA 4.0).	☒	☐
The source code is available under a GNU General Public License (GPLv3).	☒	☐
The collected data is available under a Creative Commons License (CC BY-SA 4.0).	☒	☐

Koblenz, 13/10/2022 N. B. Aehle

(Place, Date) (Signature)

Note

- If you would like us to contact you for the graduation ceremony,

  please provide your personal E-mail address: nagaraj.ba75@gmail.com

- If you would like us to send you an invite to join the WeST Alumni

  and Members group on LinkedIn,
  please provide your LinkedIn ID : /nagaraj-bahubali-asundi-77274190/

Abstract

The ambiguity of author names in digital libraries leads to incorrect document retrieval and ultimately to incorrect attribution to authors. Name disambiguation is still a hot research topic due to the challenges it presents. To address this problem, this thesis introduces *AND-MCGC - **A**uthor **N**ame **D**isambiguation using **M**arkov **C**hain-based **G**raph **C**lustering*, a method based on Markov Chain Monte Carlo sampling to generate disjoint clusters of papers such that each cluster contains papers belonging to a single real-world author. The method constructs a network of papers and repeatedly modifies the topology of this network to generate sub-graphs with homogeneous papers. In modifying the topology of the network, several discriminative features are used, such as the authors' research area, the pattern of coauthorship, the topical publication patterns over the years, and affiliations. The proposed approach achieves an F1 score of 50.29%, outperforming one of the baselines used for comparison. Extensive experiments were conducted to identify the features that contribute most to name disambiguation. The best results were obtained when all features were combined.

Contents

1 Introduction

1.1 Motivation

Over the years, the importance of Digital Libraries(DLs) in academia has grown tremendously due to a variety of factors, including cuts in budgets for traditional libraries, virtually unlimited storage at much lower costs, ease of use, 24/7 availability, and advances in information technology [1, 2, 3]. The academic community often relies on these DLs such as DBLP[1], CiteSeerX[2], Google Scholar[3], Microsoft Academic[4], etc., to search for the latest work in their respective research domain. DLs also provide useful analysis to support decision-making by funding agencies and academic institutions when awarding grants and promoting individuals [4]. Therefore, it is critical for DLs to maintain the quality of information catered to their users. However, for a variety of reasons, DLs often fail to maintain consistent and accurate information in their databases [5].

The inconsistencies in DLs lead to several problems [6], of which name disambiguation is one. Name disambiguation refers to the process of resolving the ambiguity regarding the authorship of documents whose metadata is stored in a DL. This occurs due to the fact that many DLs acquire their content in a dynamic and decentralized manner (e.g., using the Open Archives Protocol for Metadata Harvesting [7]), and lack of specific standard formats and processes shared among sources, leading to various types of author name inconsistencies [6]. According to Christen [8] and Ferreira et al. [9], the main sources of error in DLs are the typographical, imperfect citation-gathering software, disparate citation formats, ambiguous author names, decentralized content creation, etc. Among these sources of error, ambiguous author names have received considerable attention from the research community due to their inherent difficulty.

Author name ambiguity arises when the same author appears in different name variants (*synonymy*) and different authors share the same name (*homonymy*). To better understand these challenges, I have extracted citations from the bibliography of 4 different papers, as shown in Table 1. *J. Lee* and *JangMyung Lee* from sources **a** and **b** refer to the same author from Pusan National University, South Korea (*synonyms*). On the contrary, *W. Chen* in **c** refers to Wei Chen, Principal Researcher at Microsoft Research Asia, while *W. Chen* in **d** refers to Dr. Wei Chen, Professor of Engineering Design at Northwestern University (*homonyms*).

Homonymy and *synonymy* affect the quality of scientific data gathering and consequently lead to incorrect identification and attribution to authors. Recently, many approaches have been proposed to solve this problem [10, 11, 12, 13], which aim to solve the ambiguity of authors by clustering papers such that every cluster contains papers written by only a single real-world author. Clustering is achieved by using

[1] `https://dblp.org/`
[2] `http://citeseerx.ist.psu.edu/`
[3] `https://scholar.google.com/`
[4] `https://academic.microsoft.com/home`

Issue Type	Source	Citations
Synonymy	**a**	T. Jin, **J. Lee**, and H. Hashimoto, "Internet-based obstacle avoidance of mobile robot using a force-reflection," in Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems, (Sendai, Japan), pp. 3418– 3423, October 2004.
	b	TasSeok Jin, **JangMyung Lee**, and Hideki Hashimoto, "Internet-based obstacle avoidance of mobile robot using a force-reflection," IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3418-3423. 2004.
Homonymy	**c**	**W. Chen**, L. V. S. Lakshmanan, and C. Castillo, Information and Influence Propagation in Social Networks. San Rafael, CA, USA:Morgan & Claypool Publishers, 2013.
	d	Xiong, Y., **W. Chen**, D. Apley, and X. Ding. 2007. A non-stationary covariance-based Kriging method for metamodelling in engineering design. International Journal for Numerical Methods in Engineering 71:733–756.

Table 1: Real-world examples of author name ambiguity.

similarity metrics on various features or by learning the representation of the papers using network embeddings. However, these methods still have some drawbacks, as they either do not make the best use of all available factors, use less efficient similarity measures, or use non-scalable methods.

Zhang et al. [10] aim to learn the relationship representation or topology of the paper network by employing network embeddings. However, the focus is limited to learning the topology of the graph, and the domain of papers is not considered. Wang et al. [11] use generative adversarial modules to learn both content and relation representations for papers, but the strategy used to generate homogeneous papers (belonging to the same real-world author) requires computing the similarity between each pair of them and is therefore not scalable. Zhang et al. [12] obtain similar representations for papers by using graph autoencoders, but have 2 drawbacks: Co-authorship patterns are not used when learning the topology of the network, and manual intervention is required to improve clustering results, which is not scalable for large datasets. Subramanian et al. [13] apply pairwise similarity functions on multiple features such as name, email, title, etc., before clustering. However, n-gram Jaccard overlap is used to compute title similarity, which is primitive and not

as effective as modern Natural Language Processing (NLP) methods such as BERT[5], Word2vec[6], etc.

1.2 Problem Statement and Thesis Contributions

Problem Statement

The goal is to process a set of citations containing authors and their associated papers into clusters, where each cluster contains a set of papers pertaining to a real-world author. The process of author name disambiguation is formally defined as follows:

- Let $P = \{p_1, p_2, ..., p_n\}$ be a set of n papers.
- Let $A = \{a_1, a_2, ..., a_m\}$ be a set of m real-world authors.
- A paper p_i belonging to real-world author a_j is denoted by the tuple (p_i, a_j).
- The objective of the disambiguation is to partition P into m disjoint clusters $C_1, C_2, ..., C_m$ such that $C_1 = \{p_i|(p_i, a_1)\}$, ..., $C_m = \{p_i|(p_i, a_m)\}$.

Contribution

To address the challenges mentioned in Section 1.1, this thesis aims to disambiguate author names using the Markov Chain Monte Carlo algorithm (MCMC). The contributions of this thesis are as follows:

- Introduces *AND-MCGC - Author Name Disambiguation using Markov Chain-based Graph Clustering*, a method that attempts to create clusters of homogeneous papers, where each cluster contains only the papers that are written by a single real-world author.
- Employs various factors that can be helpful in the disambiguation process, such as ethnicity of author names and similarity of authors in terms of the research area, publication pattern over the years, patterns of co-authorship and overlap of affiliation, etc.
- Conducts extensive experiments to analyze the contribution of each factor in disambiguation and validate the approach with the state of the art.
- Provides an open-source implementation of the proposed approach.

[5] https://en.wikipedia.org/wiki/BERT_(language_model)
[6] https://en.wikipedia.org/wiki/Word2vec

Thesis Outline

This thesis is organized as follows:

- Section 2 lays the technical foundation needed to understand and implement the proposed methodology. It discusses concepts such as Markov Chain Monte Carlo and the Metropolis-Hastings algorithm, from which the development of *AND-MCGC* is inspired.
- Section 3 summarises various work that has been done in the area of author name disambiguation.
- Section 4 describes the proposed approach and architecture in detail. It provides insight into the various factors used for name disambiguation.
- Section 5 provides details about the dataset used and the experimental setup required to run *AND-MCGC*. It also presents the results of the experiments under different conditions and comparisons with the baselines.
- Section 6 concludes with a summary of the main contributions of this work, its limitations, and possible improvements that can enhance the quality of the results.

2 Background

This section provides a brief overview of the concepts you will need to become familiar with to understand the approach used in this thesis. First, Markov chain and Monte Carlo simulations are discussed, which serve as a springboard for understanding Markov Chain Monte Carlo (MCMC) methods. Also, a brief description of the Metropolis-Hastings (MH) algorithm - a variant of MCMC - is given. The core algorithm used in this work is an adaptation of MH. And finally, the notion of the Word Embedding, Topic Modeling, and Kernel Density Estimation (KDE) are explained. The features generated by these methods are used to distinguish ambiguous authors.

2.1 Markov Chains

Markov chains represent a class of stochastic processes in which the future does not depend on the past, but only on the present. In other words, it is a stochastic model that describes a sequence of possible events in which the probability of each event depends only on the state that reached the previous event. A countably infinite sequence in which the chain moves the state in discrete time steps results in a Discrete-time Markov chain (DTMC). A continuous-time process is called a Continuous-time Markov chain (CTMC). For simplicity, Discrete-time Markov chains will be referred to as Markov chains in the future.

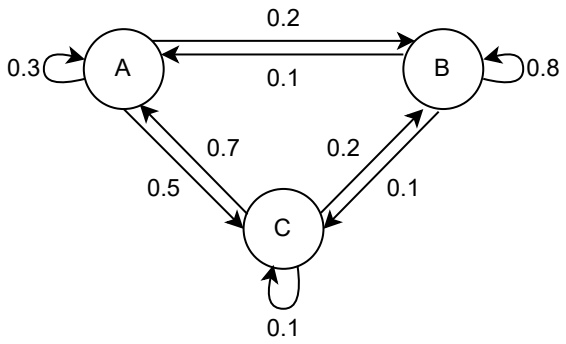


Figure 1: Diagram representing a three-state Markov process.

Mathematically, we denote the Markov chain by

$$X = (X_t)_{t \in \mathbb{N}} = (X_0, X_1, X_2, ...) \tag{1}$$

where at each instant of time t, the process takes its values in a discrete set S, such that

$$X_t \in S \quad \forall t \in \mathbb{N}$$

We refer to the discrete set S as the *state space*, which includes all possible states that the process can assume at a given time. For example, the *state space* for the Markov process in Figure 1 is $S = \{A, B, C\}$. A *trajectory* of a Markov chain is a particular set of state values for $X_0, X_1, X_2, \ldots$. For example, if X_0 = A, X_1 = B, and X_2 = C, then the *trajectory* is A, B, C up to time t = 2. More generally, if we refer to the *trajectory* $s_0, s_1, s_2, s_3, \ldots$ this means that $X_0 = s_0, X_1 = s_1, X_2 = s_2, X_3 = s_3, \ldots$.

Markov Property

The fact that the next possible state of a random process does not depend on the sequence of previous states makes Markov chains a memory-less process that depends only on the current state of a variable. This is called the *Markov property*. It means that X_{t+1} depends on X_t but does not depend on $X_{t-1}, \ldots, X_1, X_0$. We formulate the *Markov property* in mathematical notation as follows:

$$\mathbb{P}(X_{t+1} = s_{t+1} | X_t = s_t, X_{t-1} = s_{t-1}, \ldots, X_0 = s_0) = \mathbb{P}(X_{t+1} = s_{t+1} | X_t = s_t) \quad (2)$$

Transition Matrix

A Markov *transition matrix* is a square matrix that describes the probabilities of moving from one state to another in a dynamical system, also known as *transition probabilities*. Each row of the *transition matrix* contains the probabilities of transitioning from the state represented by that row to the state represented by the corresponding column. Below is an example matrix for the state transition diagram shown in Figure 1.

$$P = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1N} \\ p_{21} & p_{22} & \cdots & p_{2N} \\ \vdots & \vdots & \vdots & \vdots \\ p_{N1} & p_{N2} & \cdots & p_{NN} \end{bmatrix} = \begin{array}{c} \begin{array}{ccc} \text{A} & \text{B} & \text{C} \end{array} \\ \begin{bmatrix} 0.3 & 0.2 & 0.5 \\ 0.1 & 0.8 & 0.1 \\ 0.7 & 0.2 & 0.1 \end{bmatrix} \end{array} \begin{array}{c} \text{A} \\ \text{B} \\ \text{C} \end{array} \quad (3)$$

If $(X_0, X_1, ..., X_n)$ is a Markov chain with state space S, where S has size N, then the *transition probabilities* of the Markov chain are formally defined as:

$$p_{ij} = \mathbb{P}(X_{t+1} = j | X_t = i) \quad \text{for } i, j \in S, \quad t = 0, 1, 2, \ldots \quad (4)$$

The *transition matrix* of the Markov chain is represented as $P = (p_{ij})$, where each row should sum to 1.

$$\sum_{j=1}^{N} p_{ij} = \sum_{j=1}^{N} \mathbb{P}(X_{t+1} = j | X_t = i) = \sum_{j=1}^{N} \mathbb{P}_{\{X_t = i\}}(X_{t+1=j}) = 1 \quad (5)$$

Stationary Distributions

Let $(X_0, X_1, \ldots, X_t)$ be a Markov chain with state space $S = \{1, 2, \ldots, N\}$. Now each X_t is a random variable, so it has a probability distribution. We can write the probability distribution of X_t as an $N \times 1$ vector. For example, consider X_0. Let Π be an $N \times 1$ vector denoting the probability distribution of X_0:

$$\Pi = \begin{bmatrix} \pi_1 \\ \pi_2 \\ \vdots \\ \pi_N \end{bmatrix} = \begin{bmatrix} \mathbb{P}(X_0 = 1) \\ \mathbb{P}(X_0 = 2) \\ \vdots \\ \mathbb{P}(X_0 = N) \end{bmatrix} \tag{6}$$

Here, Π is the probability distribution for the states at time $t = 0$. With each time step, the distribution of the states changes - some states become more probable and others less probable, and this is dictated by P. The distribution of the states at time t is given by:

$$X_t = \Pi^T P^t \tag{7}$$

The *stationary distribution* of a Markov chain describes the distribution of X_t after a sufficiently long time such that the distribution of X_t no longer changes. Π is the *stationary distribution* if it has the property:

$$\Pi^T = \Pi^T P \tag{8}$$

Not all Markov chains have a *stationary distribution*, but for some classes of probability transition matrices (those defining *ergodic* Markov chains), the existence of a *stationary distribution* is guaranteed. A Markov chain is called an *ergodic* chain if it is possible to go from any state to any state (not necessarily in one go).

2.2 Monte Carlo Method

The Monte Carlo Method (MCM) is a statistical sampling technique that has been successfully applied to a variety of scientific problems over the years [14]. It was invented by John von Neumann and Stanislaw Ulam during World War II to improve decision-making under uncertain conditions. The Monte Carlo Method, also known as Monte Carlo simulation, is a general term for a vast range of computational algorithms that rely on repeated random sampling to obtain numerical results. The underlying concept is to use randomness to solve problems that could, in principle, be deterministic. Most of these algorithms have 4 steps in common [15]:

1. set up a predictive model identifying both the dependent variables and the independent variables, as well as the respective domain of possible input values
2. specify the probability distributions of the independent variables to generate the inputs randomly over the domain

3. compute the output based on the randomly generated inputs
4. repeat the experiment N times so that enough results are collected to form a representative sample from the nearly infinite number of possible combinations

Monte Carlo simulation is essentially the generation of random objects or processes using a computer. These objects may arise naturally in the context of modeling a real system, such as a complex road network or the evolution of the stock market. In many cases, however, random objects are introduced artificially to solve purely deterministic problems. In this case, MCM is simply a random sample of certain probability distributions. In both natural and artificial applications of Monte Carlo techniques, the idea is to repeat the experiment many times to obtain many quantities of interest using the *Law of Large Numbers* [16].

2.3 Markov Chain Monte Carlo

MCMC sampling provides a class of algorithms for systematic random sampling from high-dimensional probability distributions [17]. These methods have gained enormous importance in the last two decades [18, 19, 20, 21, 22], attracting intense research interest and giving new impetus to Bayesian statistics. Markov chain sampling methods originated with the work of Metropolis et al. [23], who proposed an algorithm for simulating a high-dimensional discrete distribution. This algorithm found wide application in statistical physics but was largely unknown to statisticians until the work of Hastings [24]. Hastings generalized the Metropolis algorithm and applied it to the simulation of discrete and continuous probability distributions such as the normal distribution and the Poisson distribution. Outside of statistical physics, Markov chain methods first found application in spatial statistics and image analysis [25].

The Monte Carlo method estimates the properties of distribution by examining random samples from that distribution. Unlike Monte Carlo methods, which can draw independent samples from the distribution, MCMC methods draw samples where the next sample depends on the existing sample. The Markov chain property of MCMC is that the random samples are generated by a special sequential process. Each random sample is used as a stepping stone to generate the next random sample (hence the chain). A special property of the chain is that each new sample depends on the previous one, but new samples do not depend on other samples before the previous one (this is the "Markov" property) [26].

Challenge of Probabilistic Inference

Calculating a quantity from a probabilistic model is called probabilistic inference, or more precisely *Bayesian inference* if the underlying model is a Bayesian probabilistic model. For example, we may be interested in estimating the density or other properties of the probability distribution. In Bayesian applications, the *target distribution*

or probability distribution of interest is typically the posterior distribution of the parameters when the data is available [17].

If the parameter of interest is discrete, its posterior probability is given by:

$$P(w_i|D) = \frac{P(D|w_i)P(w_i)}{P(D)} \tag{9}$$

where w_i is the different discrete possible values of the parameter, $P(w_i)$ is the prior, $P(D|w_i)$ is the likelihood and $P(D)$ is the marginal likelihood of D or is called the normalizing constant. $P(D)$ can be calculated using:

$$P(D) = \sum_{j=1}^{N} P(D|w_j)P(w_j) \tag{10}$$

If the parameter of interest is continuous, then the posterior density is given by:

$$\pi(\theta|y) = \frac{f(y|\theta)\pi(\theta)}{f_Y(y)} \tag{11}$$

where θ represents parameters of interest taking values on a continuous scale, y denotes observed sample data, $\pi(\theta)$ is the prior distribution of θ, $f(y|\theta)$ is the likelihood, and $f_Y(y)$ is the marginal likelihood. The marginal likelihood or the normalizing constant can be calculated using:

$$f_Y(y) = \int_{\infty}^{-\infty} f(y|\theta)\pi(\theta)\, d\theta \tag{12}$$

Equation (10) can be solved as the sum of a discrete distribution of random variables (w_i) according to the law of total probability. Equation (12), on the other hand, is an integral of a continuous distribution of random variables (θ) and is intractable to calculate analytically [17]. This is one of the main problems in performing Bayesian analysis.

Why MCMC

The typical solution to the above challenge is to use the Monte Carlo method to draw independent samples from the probability distribution and then repeat this process many times to approximate the desired quantity. The problem with the Monte Carlo method is that it does not work well in high-dimensions where the volume of the sample space increases exponentially with the number of parameters (dimensions). In addition, Monte Carlo sampling assumes that each random sample drawn from the target distribution is independent and can be drawn independently. This is usually not the case or impractical for inference with Bayesian structured or graphical probabilistic models [17]. On the other hand, MCMC allows for random sampling of high-dimensional probability distributions that honors the probabilistic inter-sample dependence by constructing a Markov chain that encompasses the Monte Carlo sample.

Metropolis-Hastings Algorithm

There is a huge ecosystem of MCMC variants: Gibbs sampling [27], ensemble sampling [28], Metropolis-Hastings sampling [24], parallel tempering [29], adaptive MCMC [30], Hamiltonian Monte-Carlo[31], reversible jump MCMC [32], etc. However, we will limit ourselves to MH sampling, which is crucial for understanding the algorithm used in this work.

The MH algorithm is a general MCMC method to produce sample variates from a given multivariate density [33]. It is based on a candidate generating distribution, which is used to supply a proposal value and a probability of move that is used to determine if the proposal value should be taken as the next element in the chain. The probability of a move is based on the ratio of the target distribution times the ratio of the proposal distribution. Unlike in Equation (11), since these are ratios involving target distribution, knowledge of the normalization constant of the target distribution is not required.

Let $\pi(x)$ be a target posterior distribution, where x can be any collection of parameters. To move around this parameter space, we need to formulate a proposal distribution $q(x_{i+1}|x_i)$ that gives the probability of moving to a point in the parameter space, x_{i+1} given that we are currently at x_i. The algorithm accepts a move to x_{i+1} with the following acceptance probability:

$$A(x_{i+1}|x_i) = \min(1, H) \text{ where } H = \frac{\pi(x_{i+1})\, q(x_i|x_{i+1})}{\pi(x_i)\, q(x_{i+1}|x_i)} \tag{13}$$

The term H in the equation is called Hastings Ratio. Equation (13) states that the probability of transition from x_i to the point x_{i+1} is a function of the ratio between the value of the posterior at the new point and the old point (i.e., $\pi(x_{i+1})/\pi(x_i)$), and the ratio between the transition probabilities at the new point and the old point (i.e., $q(x_i|x_{i+1})/q(x_{i+1}|x_i)$). The ratio of target posteriors ensures that the chain gradually moves to regions of high probability. And the ratio of transition probabilities ensures that the chain is not affected by preferred locations in the proposal distribution function. The algorithm is described as follows:

1. Initialize parameters x_0
2. for $i = 1$ to $i = N$:
    - Generate proposed parameters: $x_* \sim q(x_*|x_i)$
    - Sample from uniform distribution: $u \sim \mathcal{U}(0, 1)$
    - Compute Hastings ratio: $H = \frac{\pi(x_*)\, q(x_i|x_*)}{\pi(x_i)\, q(x_*|x_i)}$
    - if $u < \min(1, H)$ then $x_{i+1} = x_*$
    - else $x_{i+1} = x_i$

For each iteration, we draw the proposed parameters x_* and compute the Hastings ratio. We also draw a uniform random number u and compare it to $\min(1, H)$.

If $u < \min(1, H)$, a move is made by advancing the chain to $x_{i+1} = x_*$. If it is not, we stay at the current position x_i. In other words, the algorithm randomly tries to move in the sample space, sometimes accepting the moves and sometimes staying put. Note that the acceptance probability indicates how likely the new proposed sample is compared to the current sample, according to the distribution $\pi(x)$. If we try to move to a point that is more likely than the existing point, we will always accept the move. However, if we try to move to a point that is less likely, we will sometimes reject it, and the larger the relative decrease in probability, the more likely we will reject the new point. Thus, we will tend to stay in regions of the high density of $\pi(x)$ and only occasionally visit regions of low density.

2.4 Word Embedding

Obtaining effective representations of text words has long been a research focus in NLP and many other machine learning tasks [34, 35, 36, 37]. Unlike audio and images, which can be represented by analog or digital signals, text words cannot be processed by computers. In such cases, we often need a way to represent text words mathematically, for example, by converting each word into a binary or real-value vector [38].

A simple solution to overcome this problem would be to use a one-hot vector representation for each word in the text. Given a corpus of text, we create a vocabulary containing all the words or at least the most important ones. Furthermore, each word is represented by a binary vector whose dimension is equal to the length of the constructed vocabulary. Each word is uniquely identified by its index, as shown in Figure 2.a. This approach is combined with other techniques such as tf-idf and bag of words to accomplish the task of document classification in NLP [39]. However, this approach suffers from some shortcomings, one of which is the curse of dimensionality [35]. In language translation [40], for example, the number of words can be in the millions, resulting in a huge vocabulary. We end up creating one-hot vectors that are extremely large and sparse. This leads to inefficiencies in language modeling. Moreover, these discrete vectors are unable to capture the semantic relatedness between words.

To overcome these problems, much research has been done in learning continuous and low-dimensional vector representations, also known as *word embeddings*. As shown in this Figure 2.b., each word is represented by a vector with real values. The basic principle of word embedding is based on the *distributed hypothesis* [41], which states that words that occur in similar contexts tend to have similar semantics. Based on the approaches used, embedding techniques can be broadly divided into two groups [38]. The first is global matrix factorization methods [42, 43, 44, 45, 46] that use a low-rank approximation to decompose the large word co-occurrence matrices. These methods use the statistical information available in the text corpus to obtain *word embeddings*. The other is the sliding window-based methods, e.g. SkipGram and CBOW [47], which use shallow neural networks to perform word prediction

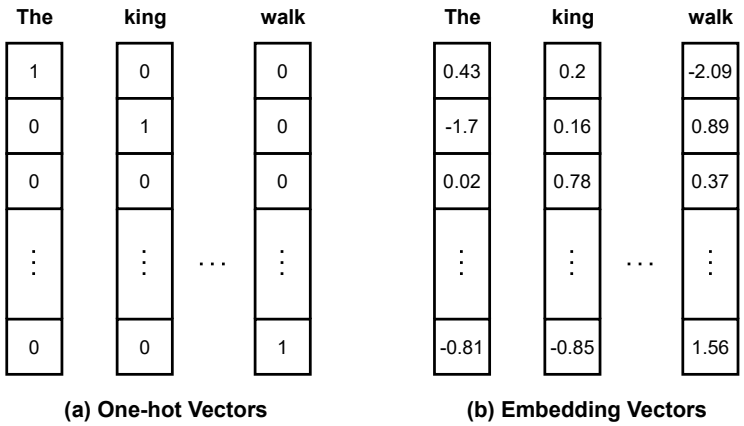


Figure 2: Word representations with vectors of binary (a) and continuous (b) values.

within a local context window [48, 49]. Here, the training corpus is prepared by a random sample of sentences, and each word in these sentences is used to predict its neighboring contextual words within a sliding window.

In the above methods, the embedding for the words is determined after training and is therefore static. As a result, they suffer from the problem of *polysemy*, i.e., the semantics of a word varies in different contexts. For example, the word "Paris" can refer to a place in France or the American pop singer "Paris Hilton". To address this problem, there has been a recent trend toward learning contextualized word representations [50, 51, 52, 53]. These methods attempt to assign embeddings to words by considering their context in the sentence so that words are represented differently in different linguistic contexts.

BERT

BERT (Bidirectional Encoder Representations from Transformers) [54] is a *transformer language model* with a variable number of encoder layers and self-attention heads, as shown in Figure 3. It uses the attention mechanism to learn the contextual relationships between words in a text. The Transformer model has two separate mechanisms - an encoder that reads the input text and a decoder that performs prediction. 15% of the words in each sequence are replaced with masked tokens before being fed into BERT. The model then attempts to predict the original value of the masked word based on the context provided by the unmasked words in the input sequence. The prediction of the output word needs:

- Addition of classification layer over the encoder output.
- Transformation of the dimension of the output vectors by multiplying them by the embedding matrix.

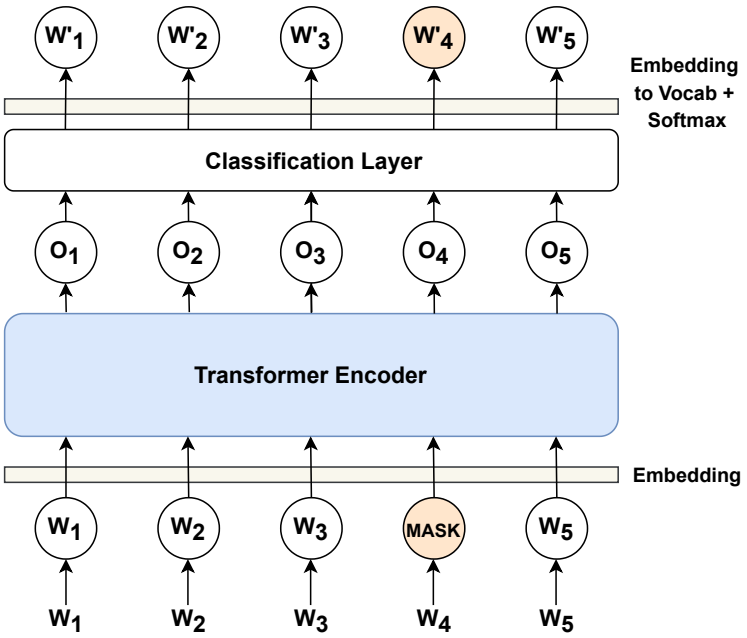


Figure 3: Architecture of BERT.

- Computation of the probability of each word in the vocabulary using Softmax.

In summary, language models help to better understand natural language and are used in various NLP tasks such as document classification [55, 56], semantic analysis [57], named entity extraction [58], question answering [59], machine translation [40], and so on. In this work, BERT was used over the titles of scientific publications to capture the authors' research area.

2.5 Topic Modeling

A topic model is a type of statistical model for finding hidden thematic patterns of words in a document collection. Topic modeling can be described as a technique for finding a collection of words, i.e., a topic, from a group of documents that represents the information in the group [60]. It can also be considered a form of text mining, a method for finding recurring word patterns in text documents [61]. Several topic modeling methods have been proposed, such as the Vector Space Model (VSM) [62], Latent Semantic Indexing (LSI) [43], Probabilistic Latent Semantic Indexing (PLSI) [63], Latent Dirichlet Allocation (LDA) [64], and Gaussian Latent Dirichlet Allocation (GLDA) [65].

Latent Dirichlet Allocation

LDA is a generative probabilistic model for collections of text documents, where each document is modeled as a finite mixture over an underlying set of topics. And each topic is in turn modeled as an infinite mixture over an underlying set of topic probabilities [64]. LDA assumes that words contain strong semantic information and that documents dealing with similar topics use a similar set of words. Latent topics are therefore discovered by identifying the groups of words in the corpus that frequently occur together in the documents. In addition, LDA assumes that documents are probability distributions of related topics and topics are probability distributions over words.

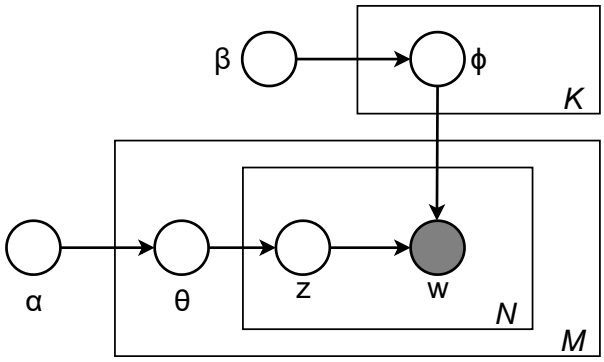


Figure 4: Plate notation of LDA.

If we have a corpus with M documents containing N words and the task is to identify K latent topics among the documents, then Figure 4 shows the visual representation of the dependencies among the LDA model parameters. Here,

- α - parameter of the Dirichlet prior on the per-document topic distributions
- β - parameter of the Dirichlet prior on the per-topic word distribution
- θ_m - topic distribution for document m
- ϕ_k - distribution of words for a given topic k
- z_{mn} - the topic for the n^{th} word in document m
- w_{mn} - specific word

Given the parameters α and β, the joint distribution of topic mixture θ, word distribution over topics ϕ, a set of K topics Z, and a set of N words W is given by:

$$P(W, Z, \theta, \phi; \alpha, \beta) = \prod_{j=1}^{M} P(\theta_j; \alpha) \prod_{i=1}^{K} P(\phi_i; \beta) \prod_{t=1}^{N} P(Z_{j,t}|\theta_j)\, P(W_{j,t}|\phi_{Z_{j,t}}) \tag{14}$$

Gaussian LDA

In traditional LDA, corpora of documents are viewed as a collection of a small number of topics, and the topics are viewed as a sparse distribution over word types [64]. Thus, a topic is a discrete distribution over a fixed vocabulary of word types. While documents can be realized as a mixture of topics, we should also expect the topics to be semantically coherent. Few studies have been conducted to assess the coherence of the words within topics produced [66, 67]. These studies suggest that the preference for semantic coherence is not encoded in the model, and even if such topics are found with coherent words, it is by chance [65]. Also, because LDA only works with a fixed number of vocabulary items, it cannot account for words outside the vocabulary that come from documents that were not originally part of the corpus. To address these issues, GLDA replaces the LDA's parameterization of topics as categorical distributions over opaque word types with multivariate Gaussian distributions on the embedding space [65]. This means that the discrete set of words present in the corpora is transformed into a continuous vector space in which each word present in the corpus is embedded as a real number.

If words are continuous vectors in an M-dimensional space and we want to derive K topics from the corpora, GLDA characterizes each topic k as a multivariate Gaussian distribution with mean μ_k and covariance Σ_k. In Bayesian probability theory, prior and posterior distributions are called conjugate distributions if the posterior distribution has a similar density to the prior probability distribution. Such a prior is then called a conjugate prior. GLDA sets conjugate priors to these values: a Gaussian centered at zero for the mean and an inverse Wishart distribution for the covariance [65]. Similar to LDA, each document is seen as a mixture of topics whose proportions are drawn from a symmetric Dirichlet prior. The steps of the generative model are as follows [65]:

1. for $k = 1$ to K
    - Draw topic covariance $\Sigma_k \sim \mathcal{W}^{-1}(\Psi, \nu)$
    - Draw topic mean $\mu_k \sim \mathcal{N}(\mu, \frac{1}{\kappa}\Sigma_k)$
2. for each document d in corpus D
    - Draw topic distribution $\theta_d \sim \mathsf{Dir}(\alpha)$
    - for each word index n from 1 to N_d
        - Draw a topic $z_n \sim \mathsf{Categorical}(\theta_d)$
        - Draw $v_{d,n} \sim \mathcal{N}(\mu_{z_n}, \Sigma_{z_n})$

In this work, GLDA was applied to the corpus of paper titles to identify authors' publication patterns.

2.6 Kernel Density Estimation

KDE, also known as Parzen's window [68], is a method for estimating an unknown probability density function from data. KDE is a non-parametric density estimator that does not require the underlying density function to be from a parametric family. KDE learns the shape of the density automatically from the data [69].

Definition: Let $X_1, X_2, \ldots, X_n$ be an independent, identically distributed random sample from an unknown distribution with density f at any given point x. Formally, KDE is defined as:

$$\hat{f}_h(x) = \frac{1}{n}\sum_{i=1}^{n} K_h(x - x_i) = \frac{1}{nh}\sum_{i=1}^{n} K(\frac{x - x_i}{h}) \tag{15}$$

where n is the number of observed data points, K is the *kernel* - a non-negative function - and $h > 0$ is a smoothing parameter called *bandwidth*. The idea is that you define a kernel function and center the kernel function on each data point. Then you sum these functions together. In other words, Kernel transforms the sharp (point) position of x_i into an interval centered (symmetrically or not) around x_i [70]. In Equation (15), $\frac{1}{n}$ normalizes the estimates, since each kernel function must have an integral that evaluates to 1 [71]. Figure 5 shows an example of a KDE transform where the red dots represent the data points, the dashed curves represent the kernel function applied to each data point, and the blue curve represents the resulting estimate.

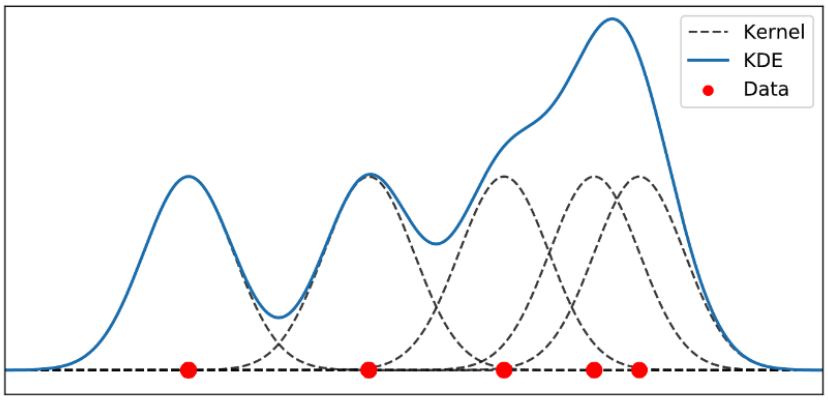


Figure 5: Construction of Kernel Density Estimator[7].

In most practical applications, symmetric kernel functions are used for kernel estimation [70], although recently asymmetric functions have been increasingly used

[7] https://github.com/tommyod/KDEpy/blob/master/docs/presentation/kde_presentation.pdf

[72, 73, 74]. For symmetric functions, the shape of a kernel is the same for all sample points, i.e., whether you go left or right of your data point, the kernel has the same value (i.e., $K(x) = K(-x)$ for each x). The shape of an asymmetric kernel, on the other hand, differs depending on the placement of the points.

For symmetric kernel functions, the choice of kernel K has little effect on the shape of the estimator [75], while the influence of the smoothing parameter h is crucial since it determines the extent of smoothing [70]. Too small a value of h may cause the estimator to display insignificant details, while too large a value of h may lead to over-smoothing of the information contained in the sample, which may subsequently obscure some important features. Figure 6 shows some of the symmetric kernel functions. In this work, Gaussian kernels are applied to the distribution of authors' publications over the years. This allows the KDEs of the two authors to be further compared for similarity.

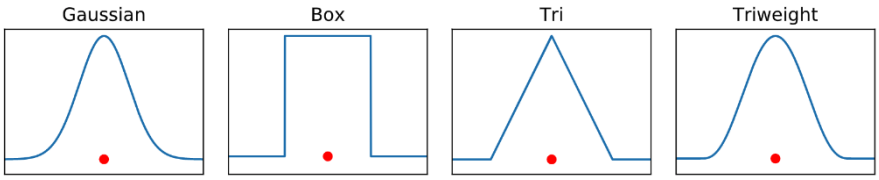


Figure 6: Variants of symmetric kernel functions[8].

[8] https://github.com/tommyod/KDEpy/blob/master/docs/presentation/kde_presentation.pdf

3 Related Work

The solutions to the name disambiguation problem broadly fall into three categories: unsupervised, supervised, and graph-based.

3.1 Unsupervised

Unsupervised methods mainly use different types of similarity metrics to divide a set of publications into different clusters, where each cluster represents the publications published by a particular individual. These similarity metrics vary in terms of the number of citation features used, such as authors, co-authors, research area, etc.

Zhang et al. [76] use features from papers combined with paper-paper relation for disambiguation. A paper relation graph is constructed using atomic clusters and further, a post-processing algorithm is used (rule-based clustering) that utilizes similarity features based on metadata information. Kim et al. [77] achieve clustering based on similarities but have introduced a new method for learning blocking schemes by using a conjunctive normal form (CNF) that reduces the number of pairwise similarity computations. Liu et al. [78] first, use the machine learning method to assign weights to the features. Based on these features, pairwise similarity scores are computed to estimate the probability that a pair of papers belong to the same author. Finally, agglomerative clustering is applied, focusing on minimizing false positive pairings. Arif et al. [79] initially create clusters based on the authors' email IDs and then apply Hierarchical Agglomerative Clustering (HAC) to merge them. This is done iteratively using publication attributes (author name, affiliation, venue, etc.) and similarity measures to arrive at the final clustering result. Wu et al. [80] also use HAC, but before clustering, various features are fused together using Dempster-Shafer theory (DST) in combination with Shannon's entropy. Khabsa et al. [81] utilize similarity features based on metadata information and citation similarity to drive the DBSCAN clustering algorithm. While Khabsa et al. [81] propose an extension of DBSCAN to handle new data points, Qian et al. [82] periodically perform Incremental Author Disambiguation (IncAD) to determine whether each new record can be assigned to an existing cluster or a new cluster.

3.2 Supervised

In a supervised setting, each author is considered a class and the goal is to classify citations into author classes. Methods such as Support Vector Machine (SVM), logistic regression, Naive Bayes, and deep learning are used.

Gao and Yan [83] label authors using ORCID[9] and apply logistic regression and SVM to classify papers. Han et al. [84] use citations and find the author class with the maximal posterior probability of producing the citation by employing Naive Bayes. Sun et al. [85] manually label authors as ambiguous or not and use two

[9] https://orcid.org/

classes of features for training a binary classifier. The first is a heuristic based on the percentage of citations gathered by the top name variations for an author. The second feature class utilizes crowd-sourced data to detect ambiguity at the topic level. Hourrane et al. [86] employ a corpus-based approach that uses deep learning word embeddings to compute citation similarities, while Muhammad et al. [87] propose a system called DEEPER to achieve the same. DEEPER is implemented using a combination of Bidirectional Recurrent Neural Network (BRNN) and Long Short Term Memory (LSTM) to generate a distributed representation for citations that can be used to determine similarities between them. Ganesh et al. [88] propose Author2Vec, a model that uses deep learning to overcome the link sparsity problem faced by models like DeepWalk [89].

3.3 Graph-Based

Graph-based methods rely on the topology of the citation network, which consists of different types of nodes, to learn the similarities between papers. In such graphs, the nodes represent citation attributes such as author, title, publication, venue, etc., while the edges represent the relationships between them. Each of these methods differs in terms of the techniques used to learn the network representations, also known as network embedding.

Deepwalk [89] learns vertex representations by modeling a stream of short random walks using word2vec [47]. These representations are latent features of vertices that capture neighborhood similarity. These latent representations encode vertex relationships in a continuous vector space with a relatively small number of dimensions. Node2Vec [90] applies Breadth First Search (BFS) and Depth First Search (DFS) search to random walk in order to extract topology information. Deepwalk and Node2vec learn vertex representations by focusing only on first-order proximity, i.e., the local pairwise proximity between two vertices. However, LINE [91] improves on this by exploring the second-order proximity between vertices, which is determined not by the observed tie strength but by the shared neighborhood structures of the vertices. Wang et al. [11] introduce generative adversarial modules using Doc2vec [92] to represent content and Node2vec [90] to learn the topology of a Heterogeneous Information Network (HIN). The generative adversarial modules consist of generative and discriminative modules. The generative module aims to find possible homogeneous papers from the HIN, while the discriminative module tries to distinguish the generated pseudo-positive papers. In this way, the reward from the discriminative module guides the exploration of the generative module to select homogeneous papers. Zhang et al. [10] use network embedding on anonymized graphs that protect the privacy of authors, and group documents into disjoint sets. Zhang et al. [12] transform the feature set with Word2Vec [47] and generate new embeddings by applying the graph auto-encoder. Later, HAC is used for partitioning and the results are improved by manual feedback.

4 Approach

4.1 Overview

This thesis introduces *AND-MCGC - Author Name Disambiguation using Markov Chain-based Graph Clustering*, a method for generating clusters of homogeneous papers by leveraging various factors such as ethnicity of author names, the similarity of authors in terms of the research area, publication pattern over the years, the pattern of co-authorship, etc. *AND-MCGC* transforms citation strings into a graph with nodes representing various features, and iteratively merges or splits these groups of nodes until clusters of homogeneous papers are generated. The entire approach used is shown in Figure 7.

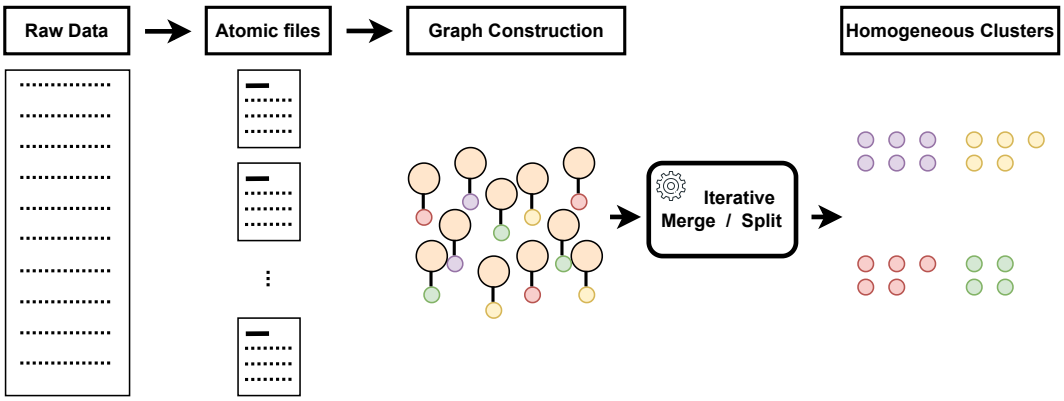


Figure 7: Architecture of *AND-MCGC*.

First, the citations are collected and preprocessed, as shown in Figure 8. Each citation string is parsed to extract the features of the paper. These citations are then divided into author namespaces (atomic name). A namespace is formed by the initials of the first name and the full last name of the author. For example, Chen Li and Cheng Li belong to C Li and we call C Li the atomic name. The authors who are in the same namespace are considered potential candidates for disambiguation. Each citation string is transformed into a graphlet (subgraph) consisting of atomic nodes and paper nodes. The edges are undirected and connected between the atomic node and the paper node. An atomic node has a single property, namely the atomic name. The paper node has multiple properties such as title, year, venue, and co-authors. The co-author in turn has 2 properties, namely the co-author's name, and the co-author's affiliation. The goal is to iteratively merge or split the graphlets so that the overall graph reaches an *ideal state*. The *ideal state* of the graph is a graph consisting of homogeneous graphlets, where each homogeneous graphlet contains a set of paper nodes that belong to a real-world author. In other words, realize a particular topology of the graph that reflects the real-world or actual distribution of author-paper relationships.

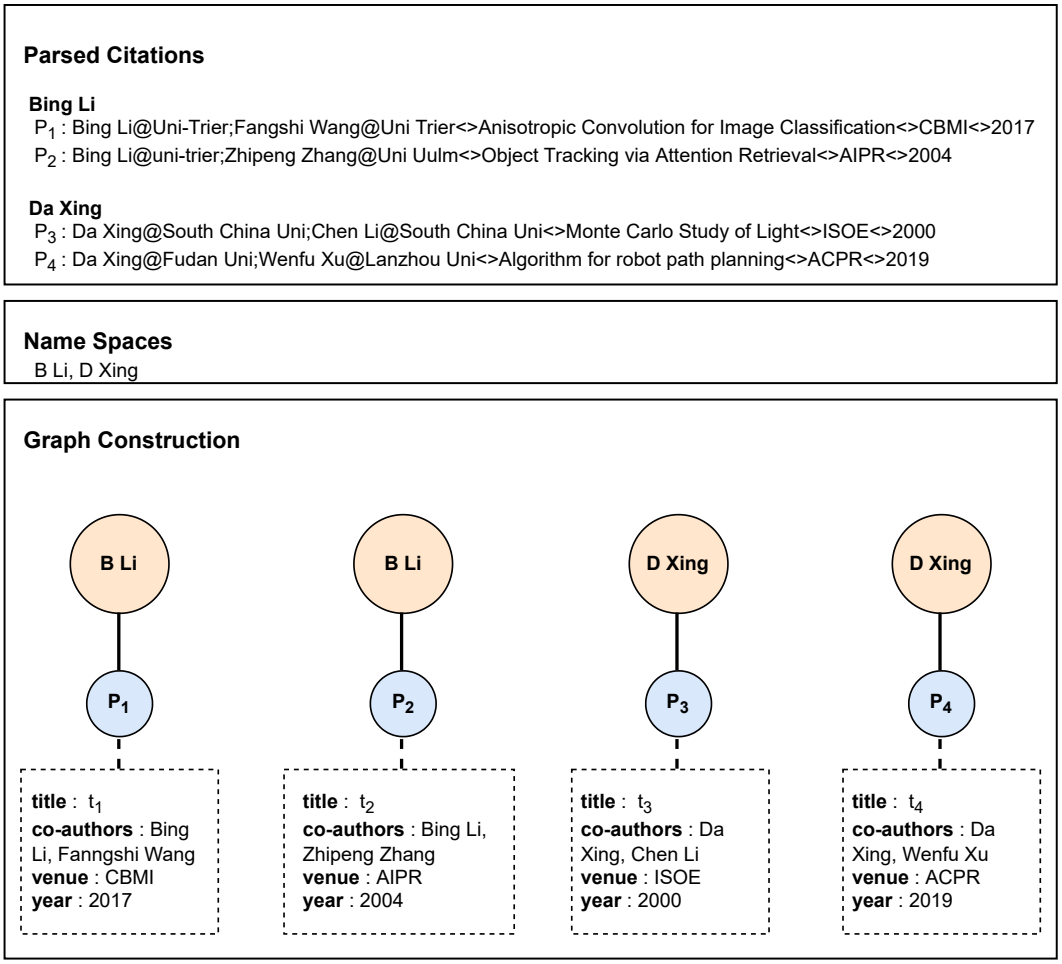
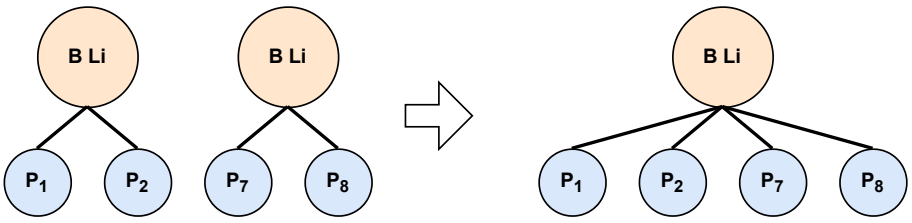


Figure 8: Pre-processing of citations.

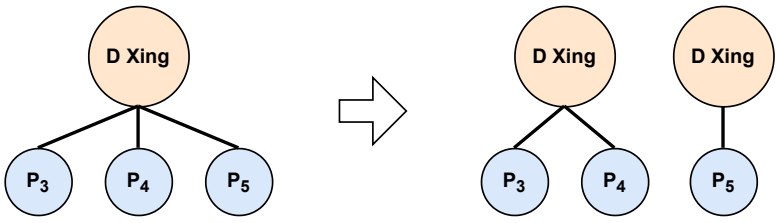
To achieve this, the MH algorithm, a variant of the MCMC method, is used in this thesis. In the MH algorithm for estimating the target distribution, we start with an initial random variable and propose subsequent variables in the form of a Markov chain. This means that the next random variable that is sampled/proposed depends on its predecessor. Each of these random variables is either *accepted* or *rejected* by being evaluated against certain *acceptance criteria*. The *acceptance criteria* must be defined in such a way that they capture the characteristics of the target distribution. Thus, they guide the sampling process so that we reach a state in which our sampling distribution reflects the target distribution. In other words, we sample random variables from a target distribution to construct a sampling distribution where the decision of whether to include the random variable in the sampling distribution is based on *acceptance criteria*. In our case, the target distribution is the *ideal state* of the graph (as defined earlier). To estimate this, we start with the initial state, where we have as many graphlets as the number of citations. Then we iteratively propose

the next state of the graph until the *ideal state* of the graph is reached. The proposed state or topology of the graph is analogous to the random sample in the context of the MH algorithm. The proposal for the next state of the graph involves either merging or splitting the graphlets, which are defined as follows.

- **merge :** Two graphlets belonging to the same atomic names are merged into one larger graphlet. For example, in Figure 9.a. we have two graphlets with the same atomic name "B Li" associated with the paper groups $[P_1, P_2]$ and $[P_7, P_8]$. We use several criteria to check whether merging them leads to better ordering in the graph. If yes, then two independent graphlets are merged with respect to their corresponding paper groups.
- **split :** A graphlet containing multiple papers is split into two smaller graphlets. For example, in Figure 9.b. we have a graphlet with the atomic name "D Xing" associated with the paper group $[P_3, P_4, P_5]$. We again use similar criteria as before to check whether the paper group is heterogeneous (papers written by more than one real-world author) and whether the split leads to better ordering in the graph. If this is the case, the target graphlet is split according to the obtained subgroups. In this case, the subgroups are $[P_3, P_4]$, and $[P_5]$.



(a) Merging the "Bing Li" papers.



(b) Splitting the "Da Xing" papers.

Figure 9: Illustrations of merge and split actions.

In the initial phase, splitting is not possible because each graphlet contains only a single paper. To avoid this, the action of merging or splitting is determined based on the number of papers available in a graphlet.

4.2 Ethnicity Analysis

An ethnic group or ethnicity is a grouping of people who identify with each other on the basis of common characteristics that distinguish them from other groups [93]. These characteristics may include shared traditions, language, history, culture, or nation within their area of residence. The disambiguation of author names is often more problematic for researchers from non-Western cultures where personal names are traditionally less diverse (leading to homonym issues) [94]. The assumption is that when there are a number of authors belonging to different ethnic groups, the authors corresponding to the dominant ethnicity are relatively more ambiguous. And it is reasonable to spend more time on the disambiguation of these authors. For example, if the given dataset contains 60% Chinese names, 10% Arabic names, and 30% Western names, it means that ambiguity is more prevalent among the Chinese authors and we want to process more of these names for disambiguation.

Several approaches have been proposed for ethnicity extraction or classification, e.g., Ethnea [95], TextMap [96], EthnicSeer [97], etc. In this work, the API[10] provided by Ethnea was used because it provided the best results among the other methods. Ethnea uses a nearest-neighbor approach to classify ethnicity. In this approach, all instances of an author's name are identified and associated with the particular country to which they belong, and then probabilistically assigned to a set of 26 predefined ethnicities. The predominant ethnicity is then assigned as a class.

4.3 Domain Similarity

The titles of the papers in citations are always meaningful sentences and infer the domain of the scientific paper. From this, one can understand the author's field of research or interests. In this work, to encode the domain of the papers all-MiniLM-L12-v2[11], a version of BERT is used. all-MiniLM-L6-v2 is the fastest of all variants and still provides good quality in terms of embedding. For merging, a *target graphlet* is first sampled along with a *merging graphlet*. The decision to merge these graphlets is made based on the ratio of the similarity between the paper groups of the *target graphlet* and the *merging graphlet* to the similarity of the subgroups of papers within the *target graphlet*.

Let $\mathcal{P}(x)$ be a function that returns all papers in a graphlet x, $\texttt{Avg}(P)$ a function that returns the average of all embeddings of the paper set P, and $\texttt{Sim}(emb_1, emb_2)$ a function that returns the cosine similarity between the embedding vectors emb_1 and emb_2. Let g and h represent the target graphlet and the merging graphlet, respectively. Let g' and g'' be the interim splits of g such that $\mathcal{P}(g') \cup \mathcal{P}(g'') = \mathcal{P}(g)$. The ratio of domain similarity for the merge (α_{merge}) is given by:

[10] http://abel.lis.illinois.edu/cgi-bin/ethnea/search.py
[11] https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2

$$\alpha_{merge} = \frac{\alpha^{(t+1)}}{\alpha^{(t)}} = \frac{\texttt{Sim}(\texttt{Avg}(\mathcal{P}(g)), \texttt{Avg}(\mathcal{P}(h)))}{\texttt{Sim}(\texttt{Avg}(\mathcal{P}(g')), \texttt{Avg}(\mathcal{P}(g'')))} \quad \text{where } t \text{ is state of the graph} \tag{16}$$

The interim splits g' and g'' are obtained by applying k-means clustering over the embeddings of all papers in g. The intuition behind computing α_{merge} is that we want to compare the similarity of the papers in graph state t with that in state $t+1$. Here, state t represents the status quo or the current state of the graph, and state $t+1$ represents the future state in which the target and merging graphlets are combined. Based on the ratio obtained, we decide whether it is worth merging the graphlets or not.

Unlike merging, we only need a target graphlet for splitting. Therefore, a target graphlet g is selected first. Then, a paper $\hat{p}$ is selected from all papers within g such that $\hat{p}$ is distant from the rest of the papers in terms of the domain. Now 3 interim splits g', g'', and g''' are obtained by applying k-means clustering to the papers of g such that $\mathcal{P}(g') \cup \mathcal{P}(g'') \cup \mathcal{P}(g''') = \mathcal{P}(g)$. Here g''' contains the paper $\hat{p}$. The ratio of domain similarity for the split (α_{split}) is given by:

$$\alpha_{split} = \frac{\alpha^{(t+1)}}{\alpha^{(t)}} = \frac{\texttt{Sim}(\texttt{Avg}(\mathcal{P}(g')), \texttt{Avg}(\mathcal{P}(g'')))}{\texttt{Sim}(\texttt{Avg}(\mathcal{P}(g') \cup \mathcal{P}(g'')), \texttt{Avg}(\mathcal{P}(g')))} \quad \text{where } t \text{ is state of the graph} \tag{17}$$

The intuition behind the computation of α_{split} is that we want to compare the similarity within the group of papers that exclude the proposed split graphlet g''' with the similarity between $\mathcal{P}(g') \cup \mathcal{P}(g'')$ and the split graphlet g'''. In Equation (17) The numerator is larger than the denominator if the sampled split graphlet is indeed far in the embedding space (low cosine similarity) compared to the rest of the papers. If the split graphlet is closer (high cosine similarity), then the ratio gives a value smaller than 1. So, based on the ratio, we can decide whether it is worth splitting the graphlet.

4.4 Co-authorship Overlap

Authors who have already published together are more likely to continue their collaboration and publish additional papers [98]. Therefore, the use of co-author networks is well-suited for author identification [99]. In this work, the Jaccard index is used as a metric to measure the similarity between two graphlets.

Let $\mathcal{C}(x)$ be a function that returns all co-authors in graphlet x. The Co-authorship overlap between graphlets x and y is defined as:

$$\mathcal{J}(x, y) = \frac{\mathcal{C}(x) \cap \mathcal{C}(y)}{\mathcal{C}(x) \cup \mathcal{C}(y)} \tag{18}$$

The ratio of co-authorship overlap for merge (β_{merge}) is given by:

$$\beta_{merge} = \frac{\beta^{(t+1)}}{\beta^{(t)}} = \frac{\mathcal{J}(g,h)}{\mathcal{J}(g',g'')} \tag{19}$$

Here g, h, g', and g'' represent the same set of graphlets as discussed in the merging part of Section 4.3. The ratio of co-authorship overlap for split (β_{split}) is given by:

$$\beta_{split} = \frac{\beta^{(t+1)}}{\beta^{(t)}} = \frac{\mathcal{J}(g',g'')}{\mathcal{J}(g' \cup g'', g''')} \tag{20}$$

Again, g', g'', and g''' represent the same set of graphlets as discussed in the splitting part of Section 4.3.

4.5 Publication Pattern

If we have two sets of papers, we can determine whether these two sets belong to the same author by looking at the similarity of the domains, as described in Section 4.3. But an author does not necessarily publish papers belonging to a single domain. In other words, he might be interested in multiple domains, resulting in diverse publications. In such scenarios, domain similarity would be ineffective in determining whether the two sets of papers belong to the same author. To circumvent this problem, the publication pattern is used in this work.

The publication pattern models the domain of an author's publications or papers over time. In other words, we want to extract the topics from an author's papers and see how the topics are distributed over time. First, all the papers are collected from the entire dataset. Then, the titles of the papers are extracted and fed to the GLDA model to draw the latent topics in the dataset. GLDA represents each title or paper as an n-dimensional vector, with each component representing the weight of the n^{th} topic. For example, if the topic distribution of a paper is [0.06, 0.5, 0.02, 0.02, 0.4], it means that the paper is a mixture of 5 topics and is dominated by topics 2 and 5. Ideally, we would need to extract the entire text of the paper to determine the latent topics. However, this is a very computationally intensive process. Moreover, it has been empirically found that modeling only with the titles of the papers gives decent results.

Once we have the topical distribution of papers, we need to further find out what the topical distribution is over the duration of the author's publication period. For this purpose, all the papers in graphlet are collected and the publication patterns are determined per topic. In other words, we want to model the distribution of an i^{th} topic over the years independently of the j^{th} topic, given there are n latent topics with $i \leq n$, $j \leq n$, and $i \neq j$. The reason for determining the distribution per topic is so that we can more easily compare two sets of papers from two independent graphlets.

To model the distribution for an individual topic of an author, the weights (probabilities) of that particular topic are converted to whole numbers and a sample of

years corresponding to those whole numbers is generated. For example, consider the topical distribution of 4 papers from Table 2. Here papers p_1, p_2, p_3, and p_4 are published in the years 1994, 1994, 1995, and 1997 respectively. To model the author's publication pattern for Topic_1, the probabilities are converted to whole numbers using a multiplicative constant, say 100. Now a sample of years is created according to the generated whole numbers. So we have a sample with year frequencies as follows: 1994 - 80, 1994 - 30, 1945 - 1, and 1997 - 70. Once we have a sample of years proportional to the distribution of the topic, this discrete distribution is converted to a continuous distribution using KDE. The reason for using KDE is that it helps us to model the topical distribution as a smooth curve. With the continuous distribution, we can compare the similarity between two sets of papers even if the corresponding years do not match exactly.

Paper	**Year**	**Topic_1**	**Topic_2**	**Topic_3**	**Topic_4**	**Topic_5**
p_1	1994	0.8	0.02	0.02	0.06	0.1
p_2	1994	0.3	0.04	0.04	0.02	0.6
p_3	1995	0.01	0.05	0.04	0.85	0.05
p_4	1997	0.7	0.03	0.03	0.03	0.21

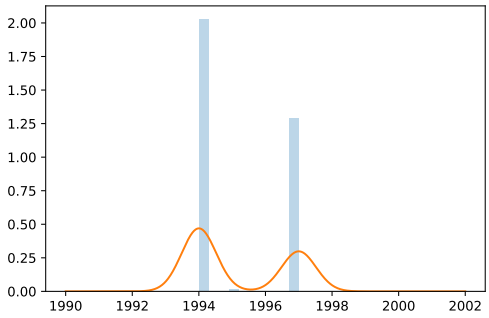
Table 2: Illustrative example for topical distribution

Figure 10 shows the transformation from discrete to continuous topical distributions by applying KDE to the data from Table 2. Figure 10.[a-e] represent the transformations for individual topics, while Figure 10.f represents all topics at once. Here, the bars correspond to the frequencies of the years and the curve is a result of KDE. Note that in Figure 10.c we get some value in the smoothed curve despite the absence of publications in 1996.

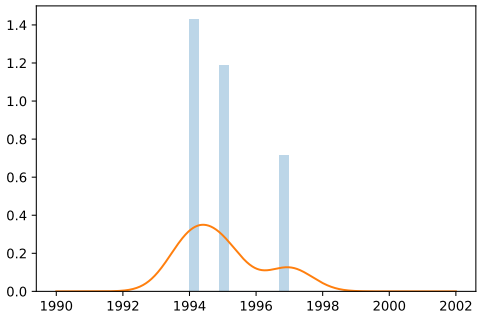
Let $t(p)$ be a function that gives the n-dimensional vector of topical distributions for paper p. The dominant topic of paper p is given by:

$$d(p) = \arg\max_x t(p) \quad \text{where } x \in \mathbb{Z} \text{ such that } 0 \leq x \leq n \tag{21}$$

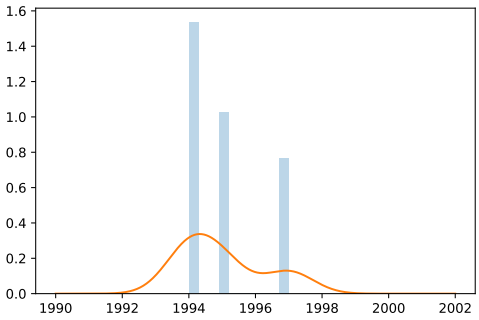
Let $\mathcal{T}^g$ be a function representing the publication pattern of all topics of papers available in the graphlet g. The publication pattern with respect to a given topic x is given by $\mathcal{T}^g_x$. Note that $t(p)$ is only the topical distribution of a single paper p, while $\mathcal{T}^g_x$ is the distribution of years (sampled according to the topical distributions) of a single topic x after applying KDE.



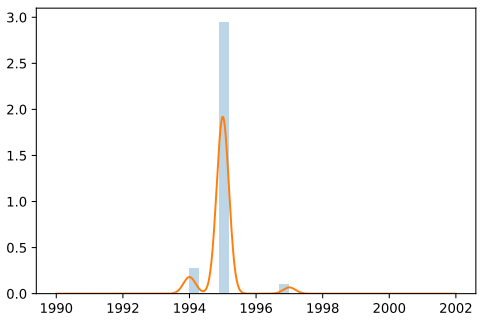
(a) KDE over Topic_1



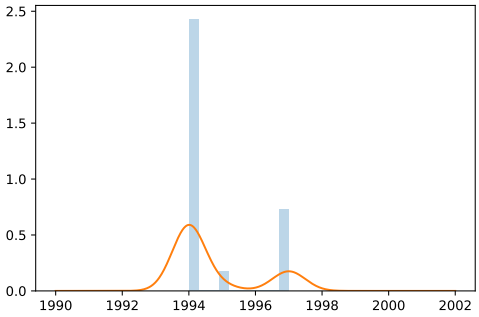
(b) KDE over Topic_2



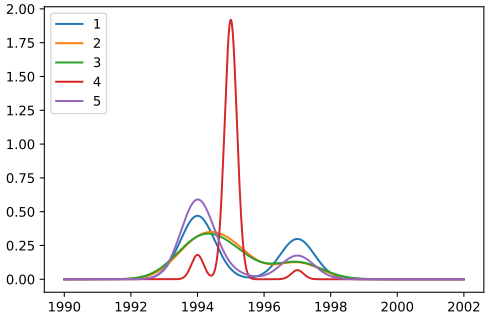
(c) KDE over Topic_3



(d) KDE over Topic_4



(e) KDE over Topic_5



(f) KDE over all Topics

Figure 10: Transformation of topical distributions using KDE.

Let $\texttt{Pdf}(\mathcal{D}, v)$ be a function that evaluates the probability density of a value v with respect to a distribution $\mathcal{D}$. In other words, it gives the probability that the value v belongs to the distribution $\mathcal{D}$. The probability that papers in graphlet h belong to graphlet g is given by:

$$\mathcal{L}(g,h) = \prod_{p \in \mathcal{P}(h)} \texttt{Pdf}(\mathcal{T}^{g}_{d(p)}, \texttt{year}(p)) \tag{22}$$

Here, all papers of the graphlet h are collected using $\mathcal{P}(h)$, and for each paper p in h, its publication year ($\texttt{year}(p)$) and dominant topic number ($d(p)$) are determined. Also, the publication pattern of the graphlet g corresponding to the dominant topic of p is obtained ($\mathcal{T}^{g}_{d(p)}$) and the probability density of the $\texttt{year}(p)$ belonging to the obtained publication pattern is calculated.

The range of the `Pdf` function is [0,1] and therefore it can return zero. This often causes the entire expression on the right-hand side of Equation (22) to be zero. To avoid this, `LogPdf` is used. The publication pattern can now be defined as follows:

$$\mathcal{LL}(g,h) = \sum_{p \in \mathcal{P}(h)} \texttt{LogPdf}(\mathcal{T}^{g}_{d(p)}, \texttt{year}(p)) \tag{23}$$

The publication patterns indeed help in modeling the similarity between groups of papers. Figure 11 shows the patterns for 3 groups of papers from the real-world dataset. Graphlets g and $h1$ contain papers of Stefan Müller from the University of Bonn, while the papers in graphlet $h2$ belong to Stefan Müller from the University of Koblenz. Here $\mathcal{LL}(g, h1)$ is -4.93, while $\mathcal{LL}(g, h2)$ is -2921.22. It can also be seen in the figure that the publication pattern of $h1$ is much closer to g than that of $h2$ to g.

The ratio of publication patterns for merge (γ_{merge}) is given by:

$$\gamma_{merge} = \frac{\gamma^{(t+1)}}{\gamma^{(t)}} = \frac{\mathcal{L}(g,h)}{\mathcal{L}(g',g'')} \tag{24}$$

Here g, h, g', and g'' represent the same set of graphlets as discussed in the merging part of Section 4.3. After the log transform the new expression becomes:

$$\begin{aligned} \ln \gamma_{merge} &= \ln \gamma^{(t+1)} - \ln \gamma^{(t)} \\ &= \mathcal{LL}(g,h) - \mathcal{LL}(g',g'') \end{aligned} \tag{25}$$

The ratio of publication patterns for split (γ_{split}) is given by:

$$\gamma_{split} = \frac{\gamma^{(t+1)}}{\gamma^{(t)}} = \frac{\mathcal{L}(g', g'')}{\mathcal{L}(g' \cup g'', g''')} \tag{26}$$

g', g'', and g''' represent the same set of graphlets as discussed in the splitting part of Section 4.3. After the log transform the new expression becomes:

$$\begin{aligned} \ln \gamma_{split} &= \ln \gamma^{(t+1)} - \ln \gamma^{(t)} \\ &= \mathcal{LL}(g', g'') - \mathcal{LL}(g' \cup g'', g''') \end{aligned} \tag{27}$$

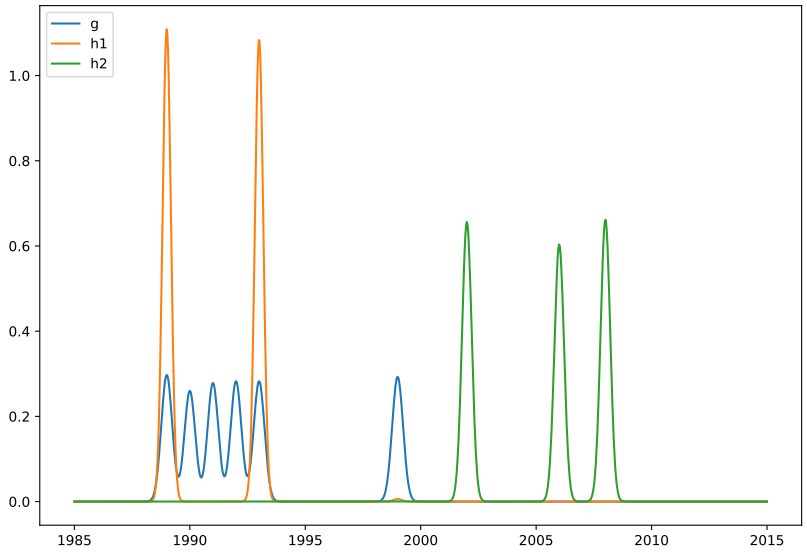


Figure 11: Similarities in Publication Patterns.

4.6 Affiliation Overlap

To disambiguate an author, we can look at his co-authors and see if there is any overlap, as mentioned in Section 4.4. Although the overlap of co-authorship can help us to find out whether both sets of co-authors belong to the same target author, there is a possibility of co-author ambiguity. In other words, there could be two different authors who also share the same set of co-author names. In such situations, we cannot rely only on the co-authors' names. We can further use the affiliation of these co-authors to see if there is any overlap and thus solve the co-author ambiguity problem.

Let $\mathcal{A}(x)$ be a function that returns all the affiliations of co-authors in graphlet x. The Affiliation overlap between graphlets x and y is defined as:

$$\mathcal{A}(x, y) = \frac{\mathcal{A}(x) \cap \mathcal{A}(y)}{\mathcal{A}(x) \cup \mathcal{A}(y)} \tag{28}$$

The ratio of affiliation overlap for merge (κ_{merge}) is given by:

$$\kappa_{merge} = \frac{\kappa^{(t+1)}}{\kappa^{(t)}} = \frac{\mathcal{A}(g, h)}{\mathcal{A}(g', g'')} \tag{29}$$

Here g, h, g', and g'' represent the same set of graphlets as discussed in the merging part of Section 4.3.

The ratio of affiliation overlap for split (κ_{split}) is given by:

$$\kappa_{split} = \frac{\kappa^{(t+1)}}{\kappa^{(t)}} = \frac{\mathcal{A}(g', g'')}{\mathcal{A}(g' \cup g'', g''')} \tag{30}$$

g', g'', and g''' represent the same set of graphlets as discussed in the splitting part of Section 4.3.

4.7 Distributions

Various distributions are used to guide the sampling in *AND-MCGC*. The distributions used in the algorithm are as follows:

- $\mathbf{D}_{ethnicities}$**(dataset) :** Distribution of the number of atomic names among the different ethnicities in the dataset. First, for each atomic name, its ethnicity is queried. Now, for each unique ethnicity, such as Arabic, Chinese, Indian, etc., the corresponding frequency of atomic name occurrence is calculated. In this way, the predominant ethnicity is sampled more frequently or the ethnicities are sampled proportionally to their occurrence in the dataset. This distribution is fixed for a given dataset and is therefore static.

- $\mathbf{D}_{atomic\_names}$**(E, t) :** Distribution of the number of graphlets over different atomic names (**AN**) belonging to a given ethnicity **E** and a state of the graph **t**. At any given state of the graph, there are different graphlets and each of these graphlets belongs to a certain atomic name **AN**. Initially, there are as many graphlets for an atomic name as there are papers available for that particular atomic name. But the number of graphlets for a particular atomic name changes over time due to split or merge operations. To sample for an atomic name, an ethnicity **E** and a state **t** of the graph are passed. The function reduces the search space to a subset of graphlets with atomic names belonging to **E**. Now, from this subset of graphlets, an atomic name is returned based on its frequency. There are as many distributions as the number of available ethnicities in the dataset. These distributions depend on the state of the graph and are therefore dynamic.

- $\mathbf{D}_{graphlets}$**(AN, t) :** Distribution of graphlets for a given atomic name **AN** in state **t**. It is a uniform distribution. In other words, the returned graphlet is simply the result of random sampling over the list of all graphlet IDs with the atomic name **AN** in graph state **t**.

- $\mathbf{D}_{actions}$**(g, t) :** Distribution of actions (merge/split) for a given graphlet **g** in state **t**. This is a Bernoulli distribution. The function guides the algorithm in choosing the appropriate action for the given graphlet. First, all graphlets that have the same atomic name **AN** as **g** are collected and the average number of papers per graphlet is calculated. If this number is larger than the number of papers in **g**, merging is preferred, otherwise splitting is preferred. Note that by the term preference, we mean a high probability for that particular action, not a hard rule that says only merge or split.

- $\mathbf{D}_{merge\_graphlets}$**(g, t) :** Distribution of the reciprocal of the number of papers over the set of graphlets excluding **g** but having the same atomic name **AN** as **g** in state **t**. In other words: If $\mathcal{G}$**(AN)** returns all graphlets with the atomic name **AN**, then a merging graphlet is selected from $\mathcal{G}$**(AN)** $\setminus$ **g** based on the reciprocal of the number of papers in those graphlets. The reason for using the reciprocal of the number of papers instead of the actual number of papers is to select those graphlets that contain relatively few papers and use them for merging.

- $\mathbf{D}_{split\_papers}$**(g, t) :** Distribution of the distance of each paper from the centroid of the embedding vectors of all papers in graphlet **g** at state **t**. The goal is to select a paper that is relatively far from others with respect to the domain. Now k-means is applied to the papers of **g** to obtain 3 interim splits. The interim split or the cluster containing the split paper $\hat{p}$ is a potential candidate for the split operation, i.e., it can be separated from the original graphlet **g**. The reason for the split operation with respect to a subset or cluster of papers containing $\hat{p}$, and not just $\hat{p}$ itself, is to enforce the reversibility of the merge operation. The reason is that the merge operation combines two sets of papers, while the split operation with respect to a single paper leads to sparsity in terms of graphlet size, i.e., graphlets with a large number of papers or a very small number of papers.

4.8 AND-MCGC

The pseudocode of *AND-MCGC* is summarized in Algorithm 1. It begins by selecting an ethnicity E, followed by an atomic name AN from E, followed by a target graphlet g. Then an appropriate action for g is selected. If the action is $merge$, another graphlet h is selected that could be a potential candidate to be merged with g. Also, two interim splits of g are obtained, namely g' and g''. The interim splits are just imaginary splits of g and have nothing to do with the $split$ operation. The deci-

sion to merge g with h is based on the acceptance criterion A_{merge}, which is defined as follows:

$$\begin{aligned} A_{merge} &= \alpha_{merge} * \beta_{merge} * \gamma_{merge} * \kappa_{merge} \\ &= \frac{\alpha^{(t+1)}}{\alpha^{(t)}} * \frac{\beta^{(t+1)}}{\beta^{(t)}} * \frac{\gamma^{(t+1)}}{\gamma^{(t)}} * \frac{\kappa^{(t+1)}}{\kappa^{(t)}} \text{ from (16), (19), (24), and (29)} \end{aligned} \tag{31}$$

After the log transform we get:

$$\ln A_{merge} = \ln \alpha^{(t+1)} - \ln \alpha^{(t)} + \ln \beta^{(t+1)} - \ln \beta^{(t)} + \ln \gamma^{(t+1)} - \ln \gamma^{(t)} + \ln \kappa^{(t+1)} - \ln \kappa^{(t)} \tag{32}$$

In (31) there is a possibility that the entire expression evaluates to zero just because one of the terms is zero. Also, division by zero may occur if the denominator of one of the terms is zero. To avoid this, the log transformation is used. Once $\ln A_{merge}$ is calculated a value u is sampled from a uniform distribution $\mathcal{U}(0, 1)$. The proposal to merge g with h is accepted if $\ln A_{merge} > \ln u$.

When the action is $split$, a paper $\hat{p}$ is selected within g that is farthest from the remaining papers with respect to the domain. Also, three interim splits of g are obtained, namely g', g'', and g'''. Here g''' contains the paper $\hat{p}$. The decision to separate g''' from g is based on the acceptance criterion A_{split}, which is defined as follows:

$$\begin{aligned} A_{split} &= \alpha_{split} * \beta_{split} * \gamma_{split} * \kappa_{split} \\ &= \frac{\alpha^{(t+1)}}{\alpha^{(t)}} * \frac{\beta^{(t+1)}}{\beta^{(t)}} * \frac{\gamma^{(t+1)}}{\gamma^{(t)}} * \frac{\kappa^{(t+1)}}{\kappa^{(t)}} \text{ from (17), (20), (26), and (30)} \end{aligned} \tag{33}$$

After log transform we get:

$$\ln A_{split} = \ln \alpha^{(t+1)} - \ln \alpha^{(t)} + \ln \beta^{(t+1)} - \ln \beta^{(t)} + \ln \gamma^{(t+1)} - \ln \gamma^{(t)} + \ln \kappa^{(t+1)} - \ln \kappa^{(t)} \tag{34}$$

The proposal to split g''' from g is accepted if $\ln A_{split} > \ln u$.

Algorithm 1 *AND-MCGC* Algorithm

Input: Set of citations(**dataset**), number of iterations(**N**)
Output: Graph with homogeneous graphlets

Preprocess citations into graphlets
$t \leftarrow 0$

repeat

  $E \sim \mathbf{D}_{ethnicities}$(dataset)
  $AN \sim \mathbf{D}_{atomic\_names}(E, t)$
  $g \sim \mathbf{D}_{graphlets}(AN, t)$
  $action \sim \mathbf{D}_{actions}(g, t)$
  $u \sim \mathcal{U}(0, 1)$

  if action $=$ *merge* **then**
    $no\_of\_interim\_splits$ = 2
    $h \sim \mathbf{D}_{merge\_graphlets}(g, t)$
    $g', g'' = \mathbf{K}_{means}(g, no\_of\_interim\_splits)$
    $A \leftarrow \ln A_{merge}(g, h, g', g'')$
    if $A > \ln u$ **then**
      Accept and merge g with h
    end if
  end if

  if action $=$ *split* **then**
    $no\_of\_interim\_splits$ = 3
    $\hat{p} \sim \mathbf{D}_{split\_papers}(g, t)$
    $g', g'', g''' = \mathbf{K}_{means}(g, no\_of\_interim\_splits, \hat{p})$
    $A \leftarrow \ln A_{split}(g', g'', g''')$
    if $A > \ln u$ **then**
      Accept and split g into $g' \cup g''$ and g'''
    end if
  end if

  $t \leftarrow t + 1$

until ($t <$ **N**)

5 Experiments and Results

This section covers the details of the dataset used, the techniques necessary to preprocess the dataset, and the experimental setup required to run the *AND-MCGC* algorithm. It also reviews several recent proposals for solving the name disambiguation problem. Finally, the Results and Discussion subsection discusses the results and summary of the various experiments performed by making parallel comparisons with the baselines.

The base version and the variants of *AND-MCGC* under different experimental setups were implemented using Python programming language, Scikit-learn, SciPy, Natural Language Toolkit (NLTK)[12], and Sentence Transformers[13]. The implementations are publicly available on GitHub[14].

5.1 Experimental Setup

Dataset

In order to compare the effectiveness of *AND-MCGC* with the state of the art, the AMiner-534K dataset [15] is used in this thesis. AMiner-534K is a knowledge graph generated from the AND benchmark dataset presented by Zhang et al. [12]. This benchmark dataset contains a subset of publications from Aminer[16] with 100 ambiguous Asian author names. Figure 12 shows the data model of the AMiner-534K knowledge graph. Figure 12 is a graffoo[17] diagram taken from Santini et al. [100] using the Diagrams.net[18] palette provided by Falco et al. [101]. For an understanding of the semantics of the graphical notations used in Figure 12, please refer to Falco et al. [101].

The AMiner-534K knowledge graphs contain 4 entities: `fabio:Expression` to represent articles, books, conference papers, and other academic works, `foaf:Agent` to represent authors, `foaf:Organization` to represent the institutions or organizations to which the authors belong, and `fabio:Journal` to represent journal venues. Each paper (`fabio:Expression`) is connected to its authors (`foaf:Agent`) by the relationship `dcterms:creator`. Thus, `fabio:Expression` has as many `dcterms:creator` relationships as the number of its co-authors. Each author has an additional `schema:affiliation` relationship with `foaf:Organization`. In addition, each paper is connected to its venue of publication (`fabio:Journal`) through the `frbr:partOf` relationship.

[12] `https://www.nltk.org/`
[13] `https://www.sbert.net/docs/pretrained_models.html#pretrained-models`
[14] `https://github.com/nagaraj-bahubali/author-name-disambiguation-using-mcmc`
[15] `https://static.aminer.cn/misc/na-data-kdd18.zip`
[16] `https://www.aminer.org/`
[17] `https://essepuntato.it/graffoo/`
[18] `https://www.diagrams.net/`

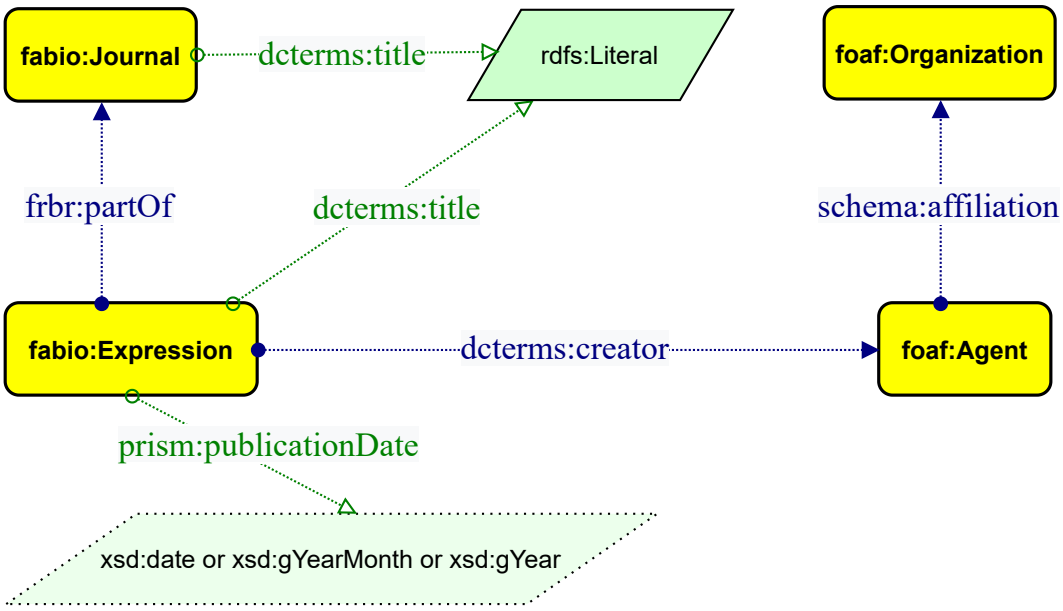


Figure 12: The data model of AMiner-534K [100].

Data Pre-Processing

First, the raw data is converted into readable citations. The mapping between the AMiner-534K and the citation attributes is shown in Table3. A citation is a string consisting of the following attributes: Author's name, list of co-authors, affiliations of co-authors, the title of the paper, venue of publication, and year of publication. Each citation follows the structure shown in Figure 13.

Entity	Attributes	Citation Attributes
`fabio:Expression`	`dcterms:title`	title
	`prism:publicationDate`	year
`fabio:Journal`	`dcterms:title`	venue
`foaf:Agent`	-	co_author
`foaf:Organization`	-	affiliation

Table 3: Mapping between entity and citation attributes.

author_id | paper_id | author_name <> co_author_1@affiliation_1; co_author_2@affiliation_2;...;co_author_n@affiliation_n <> title <> venue <> year

Figure 13: Structure of a citation string.

Citation Id	Citation String
1	1899_11618 | bo hong<>guo-xiang xing@department of neurosurgery;jian-min liu@department of neurosurgery;bo hong@department of neurosurgery<>relationship between the size and location of intracranial aneurysms and the risk of rupture<>chinese journal of cerebrovascular diseases<>2010
2	1899_11619 | bo hong<>zi-fu li@ndepartment of neurosurgery;bo hong@department of neurosurgery<>using dynact rotational angiography for evaluation and complication detection in spinal vascular diseases<>clinical neurology and neurosurgery<>2014
3	1902_11702 | bo hong<>weixuan chen@department of biomedical engineering;enhao gong@department of biomedical engineering;bo hong@department of biomedical engineering<>individualized cortical function mapping using high gamma activity<>international conference on biomedical engineering and informatics, bmei 2010<>2010
4	3536_19721 | chunyan liu<>lijuan mao@graduate school;chunyan liu@laboratory of organic optoelectronic functional materials and molecular engineering<>new route for synthesizing vo<>materials research bulletin<>2008
5	3594_20011 | chunyan liu<>chunyan liu@university of chinese academy of sciences;qing li@laboratory of aquatic biodiversity;zongbin cui@laboratory of aquatic biodiversity<>generation of an enhancer trapping vector for insertional mutagenesis in zebrafish<>plos one<>2015

Table 4: Converting raw data into citations.

Each *author name* in the citation is assigned an *author_id*. *author_id* is unique for each author in the real world. That is, if there are 10 citations belonging to 3 different authors in the real world, then the number of unique *author_ids* is 3. Each paper is also assigned a unique *paper_id*. Consider Table 4 with 5 citations. Citations 1, 2, and 3 belong to the *author_name* 'bo hong', while citations 4 and 5 belong to the *author_name* 'chunyan liu'. The author 'bo hong' from citations 1 and 2 is the same

author who works in the 'Department of Neurosurgery', so the *author_id* is also assigned the same. The author 'bo hong' in citation 3 is a different real-world author who works in the 'Department of Biomedical Engineering' and therefore has been assigned a different *author_id*. Once we get the citations, these citations are segregated based on the atomic names. In Table 4, we have 2 atomic names, namely 'B Hong' (from 'bo hong') and 'C Liu' (from 'chunyan liu'). The segregated atomic files are shown in Figures 14 and 15. Once all atomic files are created, the ethnicity of the corresponding atomic name is collected using Ethnea[19].

Atomic file: B Hong

1899_11618|bo hong<>guo-xiang xing@department of neurosurgery;jian-min liu@department of neurosurgery;bo hong@department of neuro-surgery<>relationship between the size and location of intracranial aneurysms and the risk of rupture<>chinese journal of cerebrovascular diseases<>2010

1899_11619|bo hong<>zi-fu li@ndepartment of neurosurgery;bo hong@department of neurosurgery<>using dynact rotational angiography for evaluation and complication detection in spinal vascular diseases<>clinical neurology and neurosurgery<>2014

1902_11702|bo hong<>weixuan chen@department of biomedical engineer-ing;enhao gong@department of biomedical engineering;bo hong@department of biomedical engineering<>individualized cortical function mapping using high gamma activity<>international conference on biomedical engineering and infor-matics, bmei 2010<>2010

Figure 14: Citations for the atomic name "B Hong".

[19]http://abel.lis.illinois.edu/cgi-bin/ethnea/search.py

Atomic file: C Liu

3536_19721 | chunyan liu<>lijuan mao@graduate school;chunyan liu@laboratory of organic optoelectronic functional materials and molecular engineering<>new route for synthesizing vo<>materials research bulletin<>2008

3594_20011 | chunyan liu<>chunyan liu@university of chinese academy of sciences;qing li@laboratory of aquatic biodiversity;zongbin cui@laboratory of aquatic biodiversity<>generation of an enhancer trapping vector for insertional mutagenesis in zebrafish<>plos one<>2015

Figure 15: Citations for the atomic name "C Liu".

The post-processed dataset contains a total of 35,107 citations consisting of 6,399 unique real-world authors. Figure 16 shows the top 10 atomic names with the most occurrences of ambiguous names. For example, the atomic name 'Y Zhou' is shared by 180 different real-world authors.

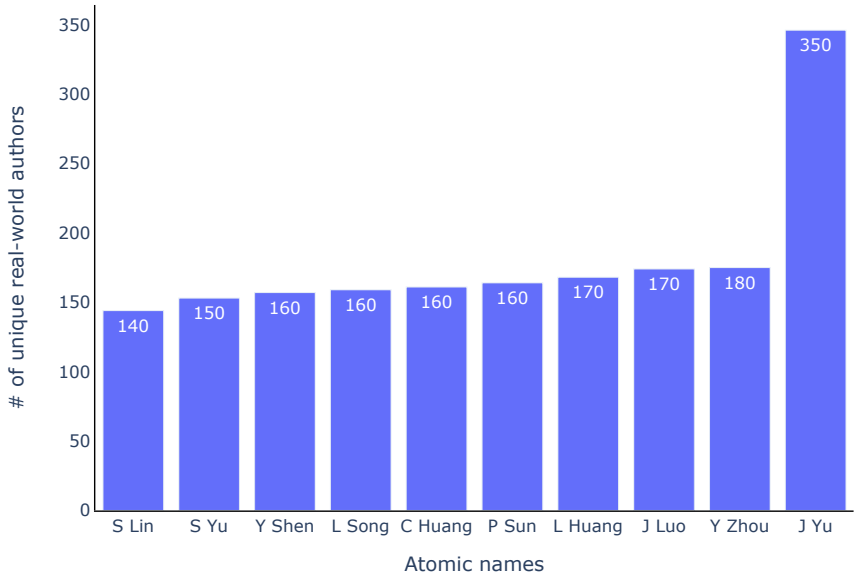


Figure 16: Top 10 atomic names in terms of ambiguity.

Latent Topic Inference

To compare the publication patterns of two graphlets, we need the prior topical distributions of the corresponding papers, as described in Section 4.5. To generate such topical distributions, GLDA is used in this work. The whole process to obtain the topical distributions is shown in Figure 17. First, the titles of all papers along with their *paper_ids* are extracted from the atomic files generated during the previous preprocessing of the data. The titles are processed to create a unique list of words called Vocab or vocabulary. The *Vocab* consists of only significant words after stop words are removed using the NLTK library. Each word in the *Vocab* is identified with a word ID. Also, for each word its embedding is generated. The *Vocab Emb* has the size vocab length $\times$ dimension of the embedding vector. Using the word IDs from *Vocab*, the titles of the papers are converted into a *Corpus* of word IDs. Finally, *Vocab Emb* and *Corpus* are fed to the GLDA model to generate topical distributions.

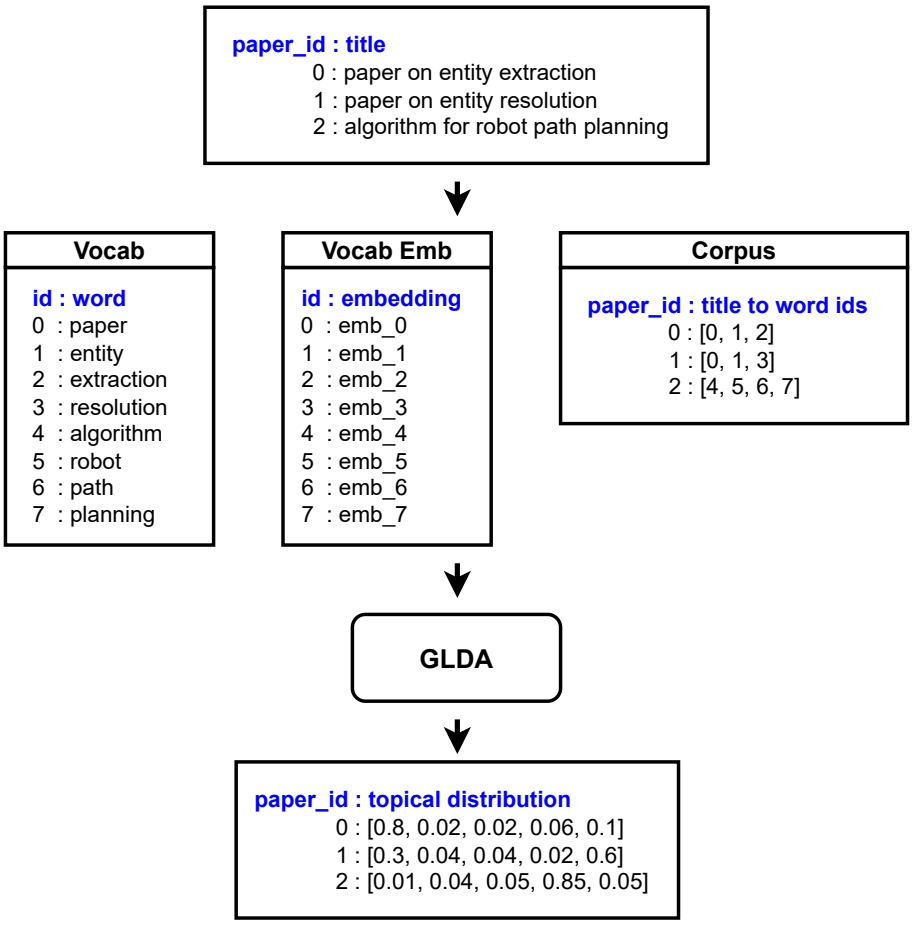


Figure 17: Input data generation for GLDA.

The authors of GLDA provide three variants of algorithms to obtain the topical

distributions, namely Naive Sampling, Faster Sampling with Cholesky Decomposition, and Faster Sampling with Cholesky + Alias Sampling. In this work, Faster Sampling with Cholesky + Alias Sampling is used because this method has been reported to be the fastest among all three in terms of running time [65]. GLDA has two hyper-parameters, namely the number of topics K and the number of iterations I. The quality of the topical distributions is measured by the average log-likelihood after each iteration. In this work, K was set to 15 and I to 20.

Hyper-parameters

To improve the results of *AND-MCGC*, various experiments are performed by fine-tuning the hyper-parameters listed below.

- **acceptance terms :** In the base version of *AND-MCGC*, four terms are used to calculate acceptance ratios (A_{merge}/A_{split}), namely Domain Similarity (α), Co-authorship Overlap (β), Publication Pattern (γ), and Affiliation Overlap (κ). The algorithm can be run with selective terms, i.e., we can choose a subset of {α, β, γ, κ} to calculate the final acceptance ratio.

- **uniform distribution :** In the base version of *AND-MCGC*, the value of u is taken from a uniform distribution $\mathcal{U}(0, 1)$ and is later compared to the acceptance ratios (A_{merge}/A_{split}) after applying the log transformation. The value of u can also be set to a constant value such as 0.7 or 1 or any value in the range 0 to 1.

- **epochs :** Number of iterations provided for the run. It can be either hard coded or set to dynamic, i.e. the number of iterations is determined based on the size (number of citations) of the atomic files.

- **early stop monitor :** It is difficult to decide how many epochs the algorithm should be run to achieve acceptable results so that it can be halted at the right time. Early stop monitoring helps to solve this problem. Early stop monitoring can be enabled by configuring two parameters: *monitor* and *significance level*. The *monitor* takes the metric like precision, recall, or f1 and the *significance level* takes float values. With this setting, the *monitor* value is observed after each epoch and checked if the new value is greater than the previous epoch value by at least the *significance level*. If so, the algorithm continues with the next epoch, otherwise, a trigger is sent to stop further execution. For example, let the *monitor* be 'f1' and the *significance level* be 0.2. We have f1 = 0.6 in epoch 10. The f1 value in the next epoch should be at least greater than 0.62 (*monitor* + *significance level*) for the algorithm to proceed to the next epoch.

- **early stop patience :** Often the first indication that the *monitor* is no longer improving is not the right time to stop. This is because it can start improving after a few more epochs. To solve this problem, an additional delay is added

in the form of successive epochs that we can wait until no more improvement is observed. The delay is added by configuring the *patience* argument of the early stop monitoring. For example, if *patience* is set to 10, the algorithm will run through 10 consecutive epochs of no improvement in the *monitor* before stopping.

- **checkpoint monitor :** Although the algorithm stops after a certain number of epochs, the state of the graph reached in the last epoch may not be the one corresponding to the best results. To avoid this, checkpoint monitoring is implemented by configuring another *monitor*. Similar to early stop monitoring, checkpoint monitoring takes one of the metrics such as precision, recall, or f1. When an improvement is detected for the *monitor* compared to the previous epoch, the state of the graph is saved. With this setting, the algorithm always saves the state of the graph at the end, which corresponds to the best results.

5.2 Baselines

To validate the performance of the proposed approach, the following state-of-the-art name disambiguation methods are considered.

- **Zhang et al. [10] :** The input data is preprocessed into three graphs: Person-Person graph - a collaboration between a pair of persons i.e. co-authors, Person-Document graph - the association of a person with a document and Document-Document similarity graph. Here, the biographical information of the authors and the keywords in the documents are not used. Network embedding is employed on these anonymized graphs and HAC is used to cluster the documents into disjoint groups.

- **GHOST [102] :** This method is based solely on the names of the co-authors. It is a graph-based framework where only attributes of co-authorship are used. The authors proposed a similarity metric to calculate the similarity between any pair of nodes in a co-authorship graph. The distance between two nodes is measured by the number of valid paths. The intermediate results of the similarity metric are clustered using affinity propagation based on message-passing techniques.

- **Louppe et al. [94] :** The authors use phonetic-based blocking strategies to form clusters and ethnicity-sensitive features are further used to learn a linkage function. Finally, a semi-supervised HAC algorithm is used to determine clusters.

- **Aminer-18 [12] :** First, documents are transformed into an embedding space using Word2Vec for features such as title, abstract, venue, co-author, etc. Triplets of documents - positive pairs with 1 and negative pairs with 0 - are formed based on whether they were authored by the same real-world author. A triplet

loss function is used to learn the global embeddings of the documents. These global embeddings are further refined by considering the local contexts within each candidate group (group of documents per author). The local linkage graph is created based on the overlap of features between documents, and the graph autoencoder is used to generate refined embeddings. Finally, HAC is used to partition the candidate groups.

- **Chen et al. [103] :** For each reference item of a name, 3 types of graphs are created, namely paper-to-paper graphs, co-author graphs, and paper-to-author graphs. The nodes in such graphs contain the attribute features and the edges contain the link features. Graph Convolutional Network (GCN) is applied to these graphs to obtain hybrid representations. Finally, HAC is applied over papers to obtain disjoint clusters.

- **Pooja et al. [104] :** The authors use an unsupervised approach that leverages both relational and non-relational aspects of the documents to generate their representations using Variational Graph Autoencoders (VGA). Here, the uniqueness of the datasets is captured by learning the co-author representations with the corresponding non-relational information such as title, abstract, location, etc. to obtain the mutual correlation between relational and non-relational features. Finally, HAC is applied to the learned representations to obtain disjoint clusters of documents.

- **Wang et al. [11] :** Document contents such as title and abstract are represented using Doc2vec [92]. The topology of academic HIN is learned using Node2vec [90]. Generative Adversarial modules are used to learn the final representations of the documents, and HAC is used to partition the documents into disjoint homogeneous groups.

- **LAND [100] :** First, the documents are grouped into blocks based on the similarity of the authors' names, using the LN-FI blocking strategy. Within each block, embeddings are obtained by learning the representative features of entities and relationships of the knowledge graph, considering the structure of the graph and the semantics embedded in the attributes of the entities, such as titles of academic papers or publication dates, and so on. HAC is applied to these embeddings to generate homogeneous clusters.

5.3 Evaluation

The final results of the approach are evaluated using pairwise measures [105] since all baselines mentioned in Section 5.2 use the same metric. Once the algorithm is run, we obtain a list of graphlets with a set of papers that can be identified by their *paper_ids*. These graphlets are nothing but the final clusters and the ground truth partitions are given by the *author_ids* of the papers. The pairwise measures attempt to capture the extent to which papers by the same author appear in the same cluster

and the extent to which papers by different authors appear in different clusters. The pairwise measures are calculated using an $r \times k$ contingency matrix. Given a clustering $\mathcal{C}$ and the ground-truth partitioning $\mathcal{T}$, the contingency matrix $\mathbf{N}$ is defined as follows:

$$\mathbf{N}(i,j) = n_{ij} = |\mathcal{C}_i \cap \mathcal{T}_j| \tag{35}$$

The count n_{ij} indicates the number of papers contained jointly in cluster $\mathcal{C}_i$ and the ground truth partition $\mathcal{T}_j$. Also, let $l_i = |\mathcal{C}_i|$ be the number of papers in cluster $\mathcal{C}_i$ and $m_j = |\mathcal{T}_j|$ be the number of papers in partition $\mathcal{T}_j$. If a pair of papers (p_i, p_j) belongs to the same cluster, it is called a positive event and if they do not belong to the same cluster, it is called a negative event. Depending on whether there is a match between the cluster labels and the partition labels, there are four possibilities that can be considered:

- **True Positives (TP) :** p_i and p_j belong to the same partition in $\mathcal{T}$, and they are also in the same cluster in $\mathcal{C}$.

  $$TP = \sum_{i=1}^{r} \sum_{j=1}^{k} \binom{n_{ij}}{2} \tag{36}$$

  This follows from the fact that every pair of papers among the n_{ij} has the same cluster label (i) and the same partition label (j).

- **False Negatives (FN) :** p_i and p_j belong to the same partition in $\mathcal{T}$, but they do not belong to the same cluster in $\mathcal{C}$. To calculate the total number of false negatives, we subtract the number of true positives from the number of paper pairs belonging to the same partition.

  $$FN = \sum_{j=1}^{k} \binom{m_j}{2} - TP \tag{37}$$

- **False Positives (FP) :** p_i and p_j do not belong to the same partition in $\mathcal{T}$, but they do belong to the same cluster in $\mathcal{C}$. The number of false positives is obtained by subtracting the number of true positives from the number of paper pairs that are in the same cluster.

  $$FP = \sum_{i=1}^{r} \binom{l_i}{2} - TP \tag{38}$$

- **True Negatives (TN) :** Finally, the number of true negatives can be determined as shown below. Here $N = \binom{n}{2}$ is the number of pairs for n papers.

  $$TN = N - (TP + FN + FP) \tag{39}$$

With this information, the pairwise measures for each atomic name are defined as follows [106]:

$$PairwisePrecision = \frac{\#PairsCorrectlyPredictedToSameAuthor}{\#TotalPairsPredictedToSameAuthor} = \frac{TP}{TP + FP} \tag{40}$$

$$PairwiseRecall = \frac{\#PairsCorrectlyPredictedToSameAuthor}{\#TotalPairsToSameAuthor} = \frac{TP}{TP + FN} \tag{41}$$

$$PairwiseF_1 = \frac{2 \times PairwisePrecision \times PairwiseRecall}{PairwisePrecision + PairwiseRecall} \tag{42}$$

The results of the algorithm are evaluated in this way for each atomic name and the overall results are calculated by taking the average for all atomic names.

5.4 Results and Discussion

In this section, several experiments are presented that differ in terms of the methodology used and the fine-tuning of the hyperparameters. Moreover, the contributions of the different parameters that assist in the process of name disambiguation are compared and contrasted. Finally, the results of all experiments are presented through a parallel comparison with the baselines.

By and large, two variants *AND-MCGC* are implemented, namely *random* and *sequential*. In the *random* variant, the sequence of operations is performed exactly as described in Algorithm 1. In the *sequential* variant, the process of sampling the ethnicity and further sampling of atomic names is skipped (steps 6 and 7 in Algorithm 1). Instead of sampling the atomic names, they are processed sequentially, i.e., the atomic files in the dataset are processed one after another. Once the result converges on a particular atomic file, the algorithm continues with the next file and so on until all files are processed.

Random Processing

Below are the hyper-parameters used in the *random* setup:

- **acceptance terms :** Used Domain Similarity (α), Co-authorship Overlap (β), Publication Pattern (γ), and Affiliation Overlap (κ) independently and also all together (all).
- **uniform distribution :** Set to sample a value in the range of $[0, 1]$, i.e. $\mathcal{U}(0, 1)$.
- **epochs :** 100,000.

Table 5 shows the summary of the experiments. The results obtained were not satisfactory for any of the combinations of hyperparameters. As can be seen in Table 5, the recall for all combinations is low, which means that the size of the predicted clusters was too small compared to the ground truths. This behavior can be confirmed in Figures 18 and 19. These figures show the distribution of cluster sizes of the predicted and ground truths. The x-axis represents the size of the clusters and the y-axis represents the number of clusters for a given cluster size.

Acceptance terms	**Precision**	**Recall**	**F1**
Domain Similarity (α)	28.29	3.66	6.48
Co-authorship Overlap (β)	35.59	2.45	4.58
Publication Pattern (γ)	92.5	2.38	4.64
Affiliation Overlap (κ)	31.1	3.0	5.47
all (α, β, γ, and κ)	28.48	4.63	7.97

Table 5: Results of *AND-MCGC* with Random Processing.

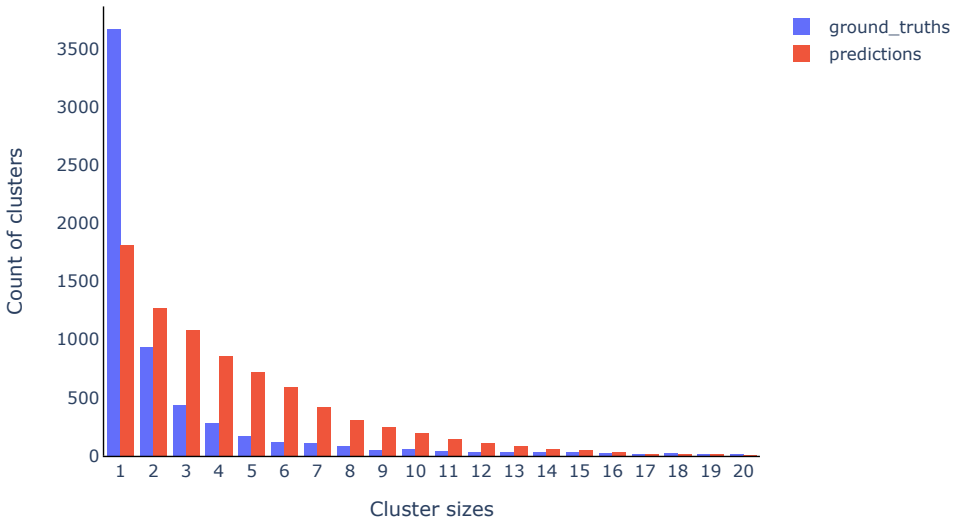


Figure 18: Distribution of cluster sizes (≤ 20).

Figure 18 shows the distribution for the cluster size ≤ 20. It can be seen that with

the exception of cluster size = 1, the number of clusters with a small size among the predictions is relatively high. Figure 19 shows the distribution for the cluster size > 20. It can be seen that the clusters with large sizes are clearly dominated by ground truths.

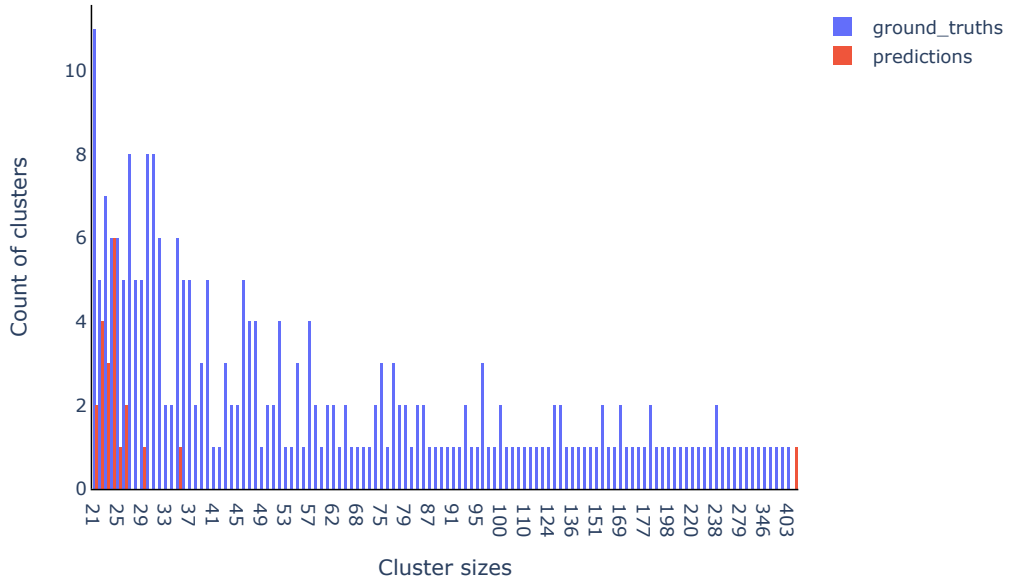


Figure 19: Distribution of cluster sizes (> 20).

The reason for yielding clusters of small size compared to the ground truth depends on how well the proposals for merging or splitting the graphlets are accepted by the algorithm. Figure 20 shows the accept/reject rates for the proposed operations. It can be noticed that about 50% of the proposals to merge and more than 60% of the proposals to split are rejected. This indicates that either the proposals are bad (i.e., propose two incorrect graphlets for merging or propose a split operation within an already homogeneous graphlet) or the computed acceptance ratio does not properly capture the inherent characteristics of the graphlets and eventually leading to rejections.

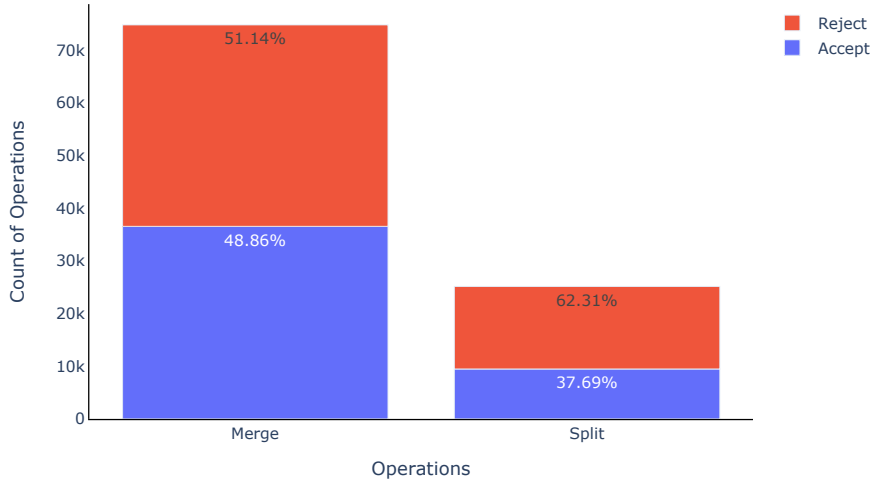


Figure 20: Accept and Reject rates for operations.

Guided Sampling

Guided sampling was used to examine the quality or correctness of proposals to merge or split. Guided Sampling uses ground truths to increase the probability of proposing the correct operations. That is, when we want to merge two graphlets, the graphlets are selected such that the author of the papers in each graphlet is the same person. Similarly, if we want to split a graphlet: A subset of papers is selected such that the subset does not share the same author as that of the original graphlet. Note that guided sampling only increases the number of correct proposals and does not ensure that all proposals are correct.

Figures 21 and 22 show the correctness of the accepted proposals for unguided (without ground truths) and guided sampling, respectively. Unguided sampling is nothing more than the same experiment performed under the settings of *random* processing. Note that among the accepted proposals for merging, the percentage of correct operations increased from 27.69% to 79.03%. On the other hand, among the proposals for splitting, it decreased from 44.14% to 27.97%. This is because, in the early phase of the algorithm, the merge operations dominate the splits since the graphlets are smaller. The graphlets became so homogeneous in the early phase that most of the later splits were incorrect. Figure23 shows the trend of the F1 score for both the guided and unguided sampling. In the unguided sampling, a maximum F1 score of 5.22% was obtained. In the guided settings, however, the F1 score increased steadily over the iterations, reaching a maximum of 25.57% which can be attributed to the improvement in the quality of the proposals.

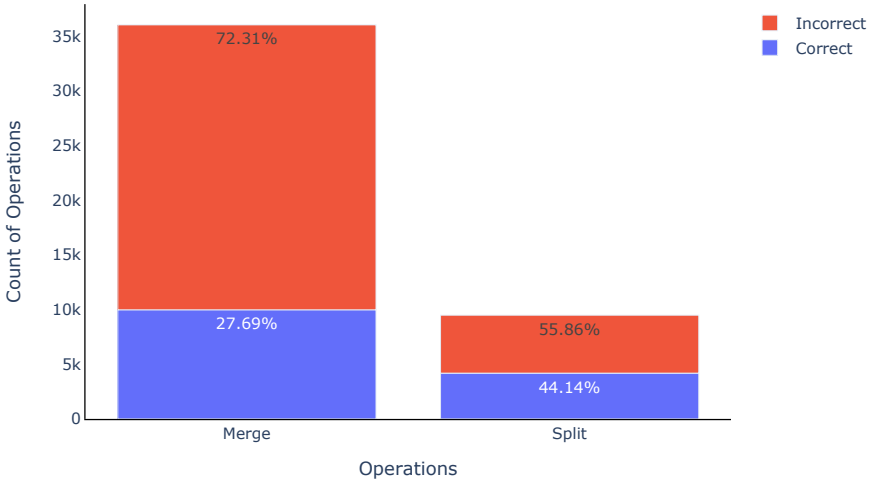


Figure 21: Correctness of accepted proposals for unguided sampling.

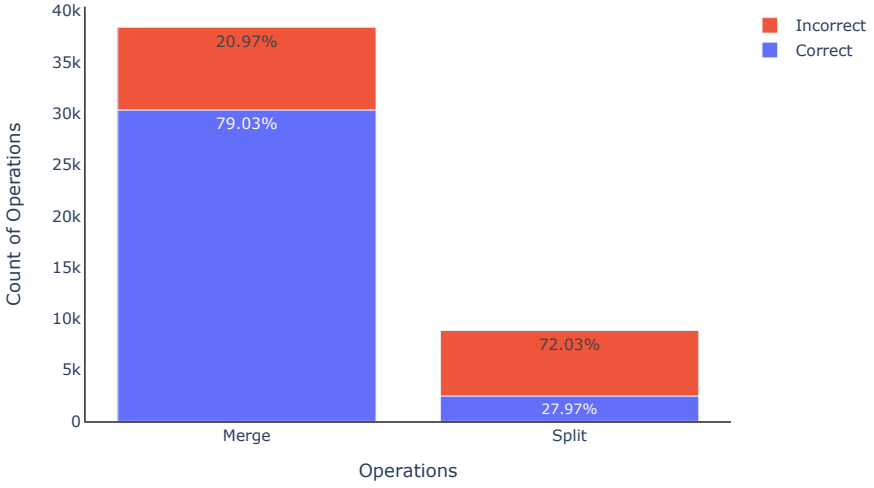


Figure 22: Correctness of accepted proposals for guided sampling.

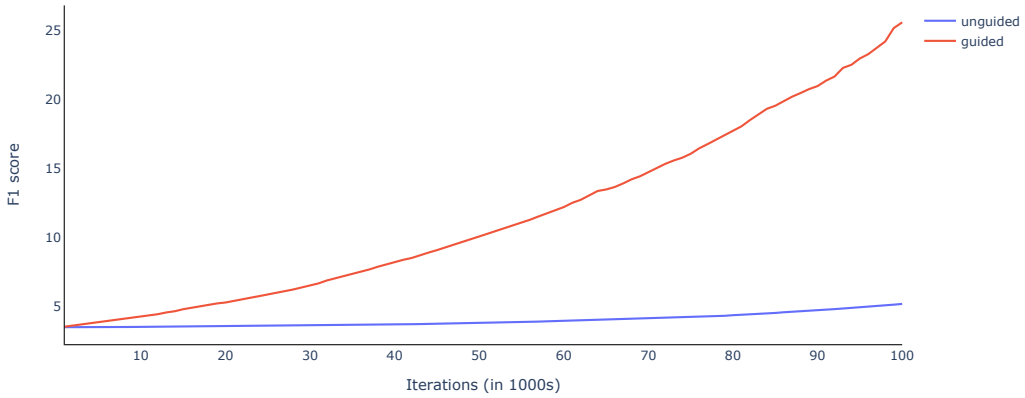


Figure 23: F1 score over iterations for unguided and guided sampling.

Single File Processing

Although the introduction of guided sampling improved the results to some extent, the F1 score was still around 25%. To dig further, the algorithm was run on a single atomic file to perform the iteration-level analysis. The single atomic file was created by sampling 30 citations from the original atomic file 'B Li' containing 5 real-world authors. Table 6 shows the results under different hyperparameter settings. With the uniform distribution set to [0,1], it was found that the number of incorrect proposals being accepted was high. This behavior was prevalent when the sampled value for u (line 10 in Algorithm 1) was smaller than 0.5. Because of the small value of u, the constraint for accepting proposals ($A > \ln u$) was weak and therefore there were a large number of incorrect merges and splits. To address this, the uniform distribution was set hard to 1 instead of sampling in the range of 0 to 1. With this setting, the behavior was opposite to what was observed for $u \sim \mathcal{U}(0, 1)$. That is, the constraint was so hard that even correct proposals were getting rejected. Further experiments were conducted by setting u to various values in the range 0.5 to 1, and 0.7 gave the best results.

For *random* processing, the number of epochs was set to 100,000, which means that the algorithm spends approximately 2.8 iterations per paper since the dataset contains 35,107 citations. With this configuration, the algorithm ran until 100,000 epochs without early stopping, as there was a continuous improvement after each epoch. This suggests that we needed even more iterations than configured to achieve convergence. To test this, during single file processing, the number of epochs was set to a high number - 1000 (more than 33 iterations per citation, since there are 30 citations). Based on the number of epochs in Table 4, we can estimate that the algorithm needs about 250 to 300 iterations to converge, i.e., about 8 to 10 iterations per cita-

tion. This proves that 2.8 iterations/citation was not enough to achieve at least good results. In summary, the best 3 results were obtained for α, γ, and all with uniform distribution set to 0.7 (it is almost the same for α).

Acceptance terms	Unif dist	Epochs	Precision	Recall	F1
Domain Similarity (α)	$[0, 1]$	111	52.10	75.65	**61.70**
	$= 1.0$	103	38.55	60.00	46.94
	$= 0.7$	126	51.95	75.47	**61.54**
Co-authorship Overlap (β)	$[0, 1]$	54	48.94	10.90	17.83
	$= 1.0$	115	34.85	34.85	34.85
	$= 0.7$	130	42.86	34.88	38.46
Publication Pattern (γ)	$[0, 1]$	196	53.33	75.47	62.50
	$= 1.0$	95	34.63	75.47	47.48
	$= 0.7$	263	82.50	62.26	**70.97**
Affiliation Overlap (κ)	$[0, 1]$	162	50.00	71.71	58.92
	$= 1.0$	197	59.29	59.29	16.76
	$= 0.7$	203	100	9.14	59.29
all (α, β, γ, and κ)	$[0, 1]$	228	49.61	57.52	53.28
	$= 1.0$	439	63.64	66.96	50.94
	$= 0.7$	109	64.29	42.19	**65.25**

Table 6: Single file results.

Sequential processing

As mentioned at the beginning of Section 5.4, in the *sequential* variant of *AND-MCGC*, the atomic files are processed one by one without sampling the ethnicity and atomic names. Based on the results obtained in single file processing, the following hyperparameters were used to obtain the results for the AMiner-534K dataset.

- **acceptance terms :** Used Domain Similarity (α), Publication Pattern (γ), and all together (all).
- **uniform distribution :** = 0.7

- **epochs :** = atomic file size $\times$ 10. Since it was observed that a single citation requires 8 to 10 iterations when processing individual files, the epochs per atomic file were set dynamically based on their size.
- **patience :** = atomic file size. Again, the number of epochs after which the algorithm should stop was not hard-coded but was determined based on the size of the corresponding atomic file.

Model	Precision	Recall	F1
GHOST [102]	81.62	40.43	50.23
Louppe et al. [94]	57.09	77.22	63.10
Zhang et al. [10]	70.63	59.53	62.81
Chen et al. [103]	65.59	69.96	65.71
Pooja et al. [104]	62.60	76.10	66.90
Aminer-18 [12]	77.96	63.03	67.79
Wang et al. [11]	82.23	67.23	72.92
LAND [100]	77.24	61.21	64.18
AND-MCGC$_{\alpha}$	47.07	54.44	43.71
AND-MCGC$_{\gamma}$	53.16	56.71	46.76
AND-MCGC$_{all}$	51.73	67.67	50.79

Table 7: Results of author name disambiguation for the AMiner-534K dataset.

Table 7 shows the performance of the different disambiguation methods along with the results from *AND-MCGC*. The best results were obtained by using all of the acceptance terms (all) with an F1 score of 50.79. In terms of precision, the proposed method did not outperform any of the baselines. This could be attributed to erroneous merging operations, i.e., merging two graphlets containing papers authored by two or more real-world authors. In terms of recall, *AND-MCGC* outperforms GHOST [102], Zhang et al. [10], Aminer-18 [12], and LAND [100]. However, the recall could have been further improved by avoiding incorrect split operations, i.e., splitting a graphlet containing papers authored by a single real-world author. As for the F1 results, *AND-MCGC* outperforms GHOST [102] by a small margin of 0.56.

6 Conclusion and Future Work

This thesis outlines the need for and challenges associated with the disambiguation of author names. It also presents *AND-MCGC*, an approach inspired by the Metropolis-Hastings algorithm and adapted to the problem of name ambiguity in digital libraries. The proposed method uses several distinguishing factors such as the authors' research area, their co-author network, topical publication patterns over a period of time, and affiliations. The results show that the authors' research area and their topical publishing patterns contribute most to disambiguation when used independently. However, better results can only be obtained when all four factors are employed. The proposed method shows promising results and outperforms at least one of the baselines.

Limitations

- The experimental setup requires that the topical distributions of the papers are available. Thus, for an end user, the integration of *AND-MCGC* is not straightforward if they only have the dataset.
- There is a degree of uncertainty when the algorithm accepts merging or splitting operations, i.e., if two graphlets are merged, the action is not persistent, and if a sufficient number of additional epochs are forced, there is a possibility that the graphlet will split into two subsets.
- The merging and splitting operations are naive to some extent. Suppose there were two graphlets, both dominated by papers by a single real-world author. When I say they are dominated, that means they will also contain a small number of papers by another author. Now, if a proposal to merge these graphlets is made and accepted, the algorithm is forced to combine them as a whole, leading to more heterogeneity. In other words, there is no mechanism to selectively choose the subset of homogeneous papers from both graphlets and combine them.
- Co-authorship overlap may not always work. In the proposed method, the names of co-authors are treated as strings, and Jaccard Overlap is used to estimate the similarity. But if we have two names like 'Bing Li' and 'B Li' in the citations, the algorithm treats them as two different authors, which reduces the overall similarity.

Future Improvements

Although the proposed approach gives promising results, there is still room for improvement.

- To avoid the additional effort in finding the thematic distribution in the papers of the dataset, a global topic modeling can be trained by collecting literature from various domains. This solves the integration problems for datasets of any type.
- Only the titles of the papers are used for topical inference. To get better topical distribution, the abstracts of the papers or even the whole text of the papers can be used.
- To estimate the domain similarity, the all-MiniLM-L6-v2 sentence transformer is used. The reason is that all-MiniLM-L6-v2 is faster than the others and still provides good quality. However, to improve the quality of sentence embedding, all-mpnet-base-v2[20] can be used.
- Although there are venues in the dataset, they are not used. The reason is that it is difficult to deal with them since their names may appear in either the full or abbreviated form. But the venues in the dataset can be profiled based on the domain of papers published there, so they serve as an additional distinguishing feature in the disambiguation process.

[20] https://huggingface.co/sentence-transformers/all-mpnet-base-v2

List of Abbreviations

AND-MCGC Author Name Disambiguation using Markov Chain-based Graph Clustering. vii, 3, 4, 20, 30, 31, 33, 34, 40, 44, 45, 50–52, 57

DL Digital Library. 1

MCMC Makov Chian Monte Carlo. 3, 5, 8–10, 21

MH Metropolis-Hastings. 5, 10, 21, 22

KDE Kernel Density Estimation. 5, 16, 17, 26

DTMC Discrete-time Markov chain. 5

CTMC Continuous-time Markov chain. 5

MCM Monte Carlo method. 7, 8

NLP Natural Language Processing. 11, 13

BERT Bidirectional Encoder Representations from Transformers. 12, 13, 23

VSM Vector Space Model. 13

LSI Latent Semantic Indexing. 13

PLSI Probabilistic Latent Semantic Indexing. 13

LDA Latent Dirichlet Allocation. 13–15

GLDA Gaussian Latent Dirichlet Allocation. 13, 15, 25, 39, 40

CNF Conjunctive Normal Form. 18

HAC Hierarchical Agglomerative Clustering. 18, 19, 41, 42

DST Dempster-Shafer theory. 18

DBSCAN Density-based spatial clustering of applications with noise. 18

IncAD Incremental Author Disambiguation. 18

SVM Support Vector Machine. 18

ORCID Open Researcher and Contributor ID. 18

BRNN Bidirectional Recurrent Neural Network. 19

LSTM Long Short Term Memory. 19

BFS Breadth First Search. 19

DFS Depth First Search. 19

HIN Heterogeneous Information Network. 19, 42

NLTK Natural Language Toolkit. 34, 39

AND Author Name Disambiguation. 34

GCN Graph Convolutional Network. 42

VGA Variational Graph Autoencoders. 42

List of Figures

List of Tables

References

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

[2] John Palfrey. Design choices for libraries in the digital-plus era. *Daedalus*, 145(1):79–86, 2016.

[3] Andrew Weiss. Examining massive digital libraries (mdls) and their impact on reference services. *The Reference Librarian*, 57(4):286–306, 2016.

[4] Ijaz Hussain and Sohail Asghar. A survey of author name disambiguation techniques: 2010–2016. *The Knowledge Engineering Review*, 32, 2017.

[5] Dongwon Lee, Jaewoo Kang, Prasenjit Mitra, C Lee Giles, and Byung-Won On. Are your citations clean? *Communications of the ACM*, 50(12):33–38, 2007.

[6] Alberto HF Laender, Marcos André Gonçalves, Ricardo G Cota, Anderson A Ferreira, Rodrygo LT Santos, and Allan JC Silva. Keeping a digital library clean: new solutions to old problems. In *Proceedings of the eighth ACM symposium on Document engineering*, pages 257–262, 2008.

[7] Carl Lagoze and Herbert Van de Sompel. The open archives initiative: Building a low-barrier interoperability framework. In *Proceedings of the 1st ACM/IEEE-CS joint conference on Digital libraries*, pages 54–62, 2001.

[8] Peter Christen. A comparison of personal name matching: Techniques and practical issues. In *Sixth IEEE International Conference on Data Mining-Workshops (ICDMW'06)*, pages 290–294. IEEE, 2006.

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

[10] Baichuan Zhang and Mohammad Al Hasan. Name disambiguation in anonymized graphs using network embedding. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1239–1248, 2017.

[11] Haiwen Wang, Ruijie Wan, Chuan Wen, Shuhao Li, Yuting Jia, Weinan Zhang, and Xinbing Wang. Author name disambiguation on heterogeneous information network with adversarial representation learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 238–245, 2020.

[12] Yutao Zhang, Fanjin Zhang, Peiran Yao, and Jie Tang. Name disambiguation in aminer: Clustering, maintenance, and human in the loop. In *Proceedings of*

the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pages 1002–1011, 2018.

[13] Shivashankar Subramanian, Daniel King, Doug Downey, and Sergey Feldman. S2and: A benchmark and evaluation system for author name disambiguation. In *2021 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, pages 170–179. IEEE, 2021.

[14] Roger Eckhardt. Stan ulam, john von neumann, and the monte carlo method. *Los Alamos Science*, 15(131-136):30, 1987.

[15] M Hénon. The monte carlo method. In *International Astronomical Union Colloquium*, volume 10, pages 151–167. Cambridge University Press, 1971.

[16] Dirk P Kroese, Tim Brereton, Thomas Taimre, and Zdravko I Botev. Why the monte carlo method is so important today. *Wiley Interdisciplinary Reviews: Computational Statistics*, 6(6):386–392, 2014.

[17] Siddhartha Chib. Markov chain monte carlo methods: computation and inference. *Handbook of econometrics*, 5:3569–3649, 2001.

[18] Stephen Brooks. Markov chain monte carlo method and its application. *Journal of the royal statistical society: series D (the Statistician)*, 47(1):69–100, 1998.

[19] George Casella, Christian P Robert, and Martin T Wells. Generalized accept-reject sampling schemes. *Lecture Notes-Monograph Series*, pages 342–347, 2004.

[20] Dani Gamerman and Hedibert F Lopes. *Markov chain Monte Carlo: stochastic simulation for Bayesian inference*. CRC press, 2006.

[21] Bradley P Carlin and Thomas A Louis. Bayes and empirical bayes methods for data analysis. *Statistics and Computing*, 7(2):153–154, 1997.

[22] Jim Albert and Siddhartha Chib. Computation in bayesian econometrics: an introduction to markov chain monte carlo. *Advances in Econometrics*, 11:3–24, 1996.

[23] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.

[24] W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. 1970.

[25] Julian Besag. Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society: Series B (Methodological)*, 36(2):192–225, 1974.

[26] Don Van Ravenzwaaij, Pete Cassey, and Scott D Brown. A simple introduction to markov chain monte–carlo sampling. *Psychonomic bulletin & review*, 25(1):143–154, 2018.

[27] Alan E Gelfand and Adrian FM Smith. Sampling-based approaches to calculating marginal densities. *Journal of the American statistical association*, 85(410):398–409, 1990.

[28] Chao Qin, Zheng Wen, Xiuyuan Lu, and Benjamin Van Roy. An analysis of ensemble sampling. *arXiv preprint arXiv:2203.01303*, 2022.

[29] Robert H Swendsen and Jian-Sheng Wang. Replica monte carlo simulation of spin-glasses. *Physical review letters*, 57(21):2607, 1986.

[30] Heikki Haario, Eero Saksman, and Johanna Tamminen. An adaptive metropolis algorithm. *Bernoulli*, pages 223–242, 2001.

[31] Simon Duane, Anthony D Kennedy, Brian J Pendleton, and Duncan Roweth. Hybrid monte carlo. *Physics letters B*, 195(2):216–222, 1987.

[32] Julian Besag, Peter Green, David Higdon, and Kerrie Mengersen. Bayesian computation and stochastic systems. *Statistical science*, pages 3–41, 1995.

[33] Antonietta Mira and Luke Tierney. On the use of auxiliary variables in markov chain monte carlo sampling. *Scandinavian Journal of Statistics*, 1997.

[34] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Interspeech*, volume 2, pages 1045–1048. Makuhari, 2010.

[35] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. journal of machine learning research, vol. 3, no, 2003.

[36] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[37] Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the 2011 conference on empirical methods in natural language processing*, pages 151–161, 2011.

[38] Qilu Jiao and Shunyao Zhang. A brief survey of word embedding and its recent development. In *2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, volume 5, pages 1697–1701. IEEE, 2021.

[39] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47, 2002.

[40] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[41] Zellig S Harris. Distributional structure. *Word*, 10(2-3):146–162, 1954.

[42] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167, 2008.

[43] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407, 1990.

[44] Paramveer Dhillon, Dean P Foster, and Lyle Ungar. Multi-view learning of word embeddings via cca. *Advances in neural information processing systems*, 24, 2011.

[45] Rémi Lebret and Ronan Collobert. Word emdeddings through hellinger pca. *arXiv preprint arXiv:1312.5542*, 2013.

[46] Peter D Turney and Patrick Pantel. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37:141–188, 2010.

[47] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[48] Andriy Mnih and Geoffrey Hinton. Three new graphical models for statistical language modelling. In *Proceedings of the 24th international conference on Machine learning*, pages 641–648, 2007.

[49] Fei Sun, Jiafeng Guo, Yanyan Lan, Jun Xu, and Xueqi Cheng. Learning word representations by jointly modeling syntagmatic and paradigmatic relations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 136–145, 2015.

[50] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[51] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacL-HLT*, pages 4171–4186, 2019.

[52] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, and Kenton Lee. Luke 440 zettlemoyer. deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 441, 2018.

[53] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32, 2019.

[54] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[55] Bill Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Third International Workshop on Paraphrasing (IWP2005)*, 2005.

[56] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. Compare, compress and propagate: Enhancing neural architectures with alignment factorization for natural language inference. *arXiv preprint arXiv:1801.00102*, 2017.

[57] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

[58] Erik F Sang and Fien De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. *arXiv preprint cs/0306050*, 2003.

[59] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

[60] Bhagyashree Vyankatrao Barde and Anant Madhavrao Bainwad. An overview of topic modeling methods and tools. In *2017 International Conference on Intelligent Computing and Control Systems (ICICCS)*, pages 745–750. IEEE, 2017.

[61] Rubayyi Alghamdi and Khalid Alfalqi. A survey of topic modeling in text mining. *Int. J. Adv. Comput. Sci. Appl.(IJACSA)*, 6(1), 2015.

[62] Gerard Salton, Anita Wong, and Chung-Shu Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[63] Thomas Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57, 1999.

[64] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[65] Rajarshi Das, Manzil Zaheer, and Chris Dyer. Gaussian lda for topic models with word embeddings. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 795–804, 2015.

[66] Jonathan Chang, Sean Gerrish, Chong Wang, Jordan Boyd-Graber, and David Blei. Reading tea leaves: How humans interpret topic models. *Advances in neural information processing systems*, 22, 2009.

[67] David Newman, Sarvnaz Karimi, and Lawrence Cavedon. External evaluation of topic models. In *in Australasian Doc. Comp. Symp., 2009*. Citeseer, 2009.

[68] Emanuel Parzen. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076, 1962.

[69] Yen-Chi Chen. A tutorial on kernel density estimation and recent advances. *Biostatistics & Epidemiology*, 1(1):161–187, 2017.

[70] Stanisław Węglarczyk. Kernel density estimation and its application. In *ITM Web of Conferences*, volume 23, page 00037. EDP Sciences, 2018.

[71] Bernard W Silverman. *Density estimation for statistics and data analysis*. Routledge, 2018.

[72] Song Xi Chen. Probability density function estimation using gamma kernels. *Annals of the Institute of Statistical Mathematics*, 52(3):471–480, 2000.

[73] Olivier Scaillet. Density estimation using inverse and reciprocal inverse gaussian kernels. *Nonparametric statistics*, 16(1-2):217–226, 2004.

[74] Xiaodong Jin, Janusz Kawczak, et al. Birnbaum-saunders and lognormal kernel estimators for modelling durations in high frequency financial data. *Annals of Economics and Finance*, 4:103–124, 2003.

[75] Peter Hall and JS Marron. On the amount of noise inherent in bandwidth selection for a kernel density estimator. *The Annals of Statistics*, pages 163–181, 1987.

[76] Lizhi Zhang and Zhijie Ban. Author name disambiguation based on rule and graph model. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 617–628. Springer, 2020.

[77] Kunho Kim, Athar Sefid, and C Lee Giles. Learning cnf blocking for large-scale author name disambiguation. In *Proceedings of the First Workshop on Scholarly Document Processing*, pages 72–80, 2020.

[78] Wanli Liu, Rezarta Islamaj Doğan, Sun Kim, Donald C Comeau, Won Kim, Lana Yeganova, Zhiyong Lu, and W John Wilbur. Author name disambiguation for p ub m ed. *Journal of the Association for Information Science and Technology*, 65(4):765–781, 2014.

[79] Tasleem Arif, Rashid Ali, and M Asger. Author name disambiguation using vector space model and hybrid similarity measures. In *2014 Seventh International Conference on Contemporary Computing (IC3)*, pages 135–140. IEEE, 2014.

[80] Hao Wu, Bo Li, Yijian Pei, and Jun He. Unsupervised author disambiguation using dempster–shafer theory. *Scientometrics*, 101(3):1955–1972, 2014.

[81] Madian Khabsa, Pucktada Treeratpituk, and C Lee Giles. Online person name disambiguation with constraints. In *Proceedings of the 15th acm/ieee-cs joint conference on digital libraries*, pages 37–46, 2015.

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

[83] Yan Gao. *Author Name Disambiguation Using Co-training*. PhD thesis, University of Windsor (Canada), 2020.

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

[85] Xiaoling Sun, Jasleen Kaur, Lino Possamai, and Filippo Menczer. Detecting ambiguous author names in crowdsourced scholarly data. In *2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing*, pages 568–571. IEEE, 2011.

[86] Oumaima Hourrane, Sara Mifrah, Nadia Bouhriz, Mohamed Rachdi, et al. Using deep learning word embeddings for citations similarity in academic papers. In *International Conference on Big Data, Cloud and Applications*, pages 185–196. Springer, 2018.

[87] Muhammad Ebraheem, Saravanan Thirumuruganathan, Shafiq Joty, Mourad Ouzzani, and Nan Tang. Distributed representations of tuples for entity resolution. *Proceedings of the VLDB Endowment*, 11(11):1454–1467, 2018.

[88] J Ganesh, Soumyajit Ganguly, Manish Gupta, Vasudeva Varma, and Vikram Pudi. Author2vec: Learning author representations by combining content and link information. In *WWW (Companion Volume)*, 2016.

[89] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

[90] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[91] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, pages 1067–1077, 2015.

[92] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196. PMLR, 2014.

[93] Kanchan Chandra. *Constructivist theories of ethnic politics*. Oxford University Press, 2012.

[94] Gilles Louppe, Hussein T Al-Natsheh, Mateusz Susik, and Eamonn James Maguire. Ethnicity sensitive author disambiguation using semi-supervised learning. In *international conference on knowledge engineering and the semantic web*, pages 272–287. Springer, 2016.

[95] Vetle I Torvik and Sneha Agarwal. Ethnea–an instance-based ethnicity classifier based on geo-coded author names in a large-scale bibliographic database. 2016.

[96] Anurag Ambekar, Charles Ward, Jahangir Mohammed, Swapna Male, and Steven Skiena. Name-ethnicity classification from open sources. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge Discovery and Data Mining*, pages 49–58, 2009.

[97] Pucktada Treeratpituk and C Lee Giles. Name-ethnicity classification and ethnicity-sensitive name matching. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 26, pages 1141–1147, 2012.

[98] Zeyd Boukhers and Nagaraj Asundi Bahubali. Whois? deep author name disambiguation using bibliographic data. *arXiv preprint arXiv:2207.04772*, 2022.

[99] Fakhri Momeni and Philipp Mayr. Using co-authorship networks for author name disambiguation. In *2016 IEEE/ACM Joint Conference on Digital Libraries (JCDL)*, pages 261–262. IEEE, 2016.

[100] Cristian Santini, Genet Asefa Gesese, Silvio Peroni, Aldo Gangemi, Harald Sack, and Mehwish Alam. A knowledge graph embeddings based approach for author name disambiguation using literals. *arXiv preprint arXiv:2201.09555*, 2022.

[101] Riccardo Falco, Aldo Gangemi, Silvio Peroni, David Shotton, and Fabio Vitali. Modelling owl ontologies with graffoo. In *European Semantic Web Conference*, pages 320–325. Springer, 2014.

[102] Xiaoming Fan, Jianyong Wang, Xu Pu, Lizhu Zhou, and Bing Lv. On graph-based name disambiguation. *Journal of Data and Information Quality (JDIQ)*, 2(2):1–23, 2011.

[103] Ya Chen, Hongliang Yuan, Tingting Liu, and Nan Ding. Name disambiguation based on graph convolutional network. *Scientific Programming*, 2021, 2021.

[104] KM Pooja, Samrat Mondal, and Joydeep Chandra. Exploiting similarities across multiple dimensions for author name disambiguation. *Scientometrics*, 126(9):7525–7560, 2021.

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

[106] Jie Tang, Alvis CM Fong, Bo Wang, and Jing Zhang. A unified probabilistic framework for name disambiguation in digital library. *IEEE Transactions on Knowledge and Data Engineering*, 24(6):975–987, 2011.