Name Disambiguation in AMiner: Clustering, Maintenance, and Human in the Loop

Yutao Zhang
Tsinghua University
yt-zhang13@mails.tsinghua.edu.cn

Fanjin Zhang
Tsinghua University
zfj17@mails.tsinghua.edu.cn

Peiran Yao
Tsinghua University
ypr15@mails.tsinghua.edu.cn

Jie Tang
Tsinghua University
jietang@tsinghua.edu.cn

ABSTRACT

AMiner[1] is a free online academic search and mining system, having collected more than 130,000,000 researcher profiles and over 200,000,000 papers from multiple publication databases [25].

In this paper, we present the implementation and deployment of *name disambiguation*, a core component in AMiner. The problem has been studied for decades but remains largely unsolved. In AMiner, we did a systemic investigation into the problem and propose a comprehensive framework to address the problem. We propose a novel representation learning method by incorporating both global and local information and present an end-to-end cluster size estimation method that is significantly better than traditional BIC-based method. To improve accuracy, we involve human annotators into the disambiguation process. We carefully evaluate the proposed framework on real-world large data and experimental results show that the proposed solution achieves clearly better performance (+7-35% in terms of F1-score) than several state-of-the-art methods including GHOST [5], Zhang et al. [33], and Louppe et al. [17].

Finally, the algorithm has been deployed in AMiner to deal with the disambiguation problem at the billion scale, which further demonstrates both effectiveness and efficiency of the proposed framework.

CCS CONCEPTS

• **Information systems** → **Information integration**; **Entity resolution**; **Clustering**;

KEYWORDS

Name Disambiguation, Entity Resolution, Clustering, Metric Learning

[1]https://aminer.org

KDD '18, August 19–23, 2018, London, United Kingdom

ACM ISBN 978-1-4503-5552-0/18/08…$15.00
https://doi.org/10.1145/3219819.3219859

ACM Reference Format:
Yutao Zhang, Fanjin Zhang, Peiran Yao, and Jie Tang. 2018. Name Disambiguation in AMiner: Clustering, Maintenance, and Human in the Loop . In *KDD '18: The 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, August 19–23, 2018, London, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3219819.3219859

1 INTRODUCTION

There are 151,671 different last names and 5,163 different first names in common use in the United States [2]. This means that if you happen to have a common first and last names, according to the birthday paradox theory [27], the probability that some other researcher in your university shares the *same name* with you is close to 100%! In practice, it is estimated that the 300 most common male names are used by more than 115 million people (taking about 78.74 percent) in the United States. This poses a big challenge to many person-centric applications such as scientific literature management and people search engine.

The problem of *disambiguating who is who* is referred to as name disambiguation, also named as entity resolution [3, 4], web appearance disambiguation [1, 11], name identification [16], and object distinction [31] from a broader viewpoint, and has been extensively studied for decades by different communities. It has many real applications, for example, matching records between enterprise databases with different schema [4], aligning protein-protein interaction networks to transfer biological knowledge across different species [30], constructing canonicalized knowledge base based on facts extracted from texts [6], and identifying users across multiple online social networks [34]. However, despite much work that has been done, the problem remains largely unsolved. Most of the aforementioned methods are more or less ad-hoc and the performance becomes unpredictable when scale up to handle real large data.

In this paper, employing AMiner as the basis for our experimental data, we explain how we deal with the name ambiguity problem with large data in an online fashion. AMiner is a free online academic search and mining system [25]. The system extracts researchers' profiles automatically from the Web [24] and integrates them with published papers after name disambiguation [23]. To date, it has collected more than 130,000,000 researcher profiles and over 200,000,000 papers from multiple publication databases, with a growth rate of over 500,000 per month. AMiner can be viewed as an author-centric search system, where one can find domain experts,

[2]http://howmanyofme.com

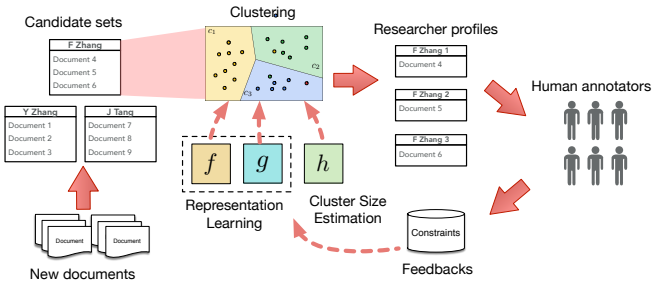


Figure 1: An overview of the author disambiguation framework in AMiner.

rising stars, collaborators, reviewers, potential readers, and so on. Clearly, name disambiguation is a cornerstone of the system.

Dealing with the name ambiguity problem in a large online system poses several unique challenges:

(1) **How to quantify the similarity between entities from different data sources?** As documents and authors are from different sources and may have no overlapping information, it is necessary to design a principled way to quantify the similarity between (different) entities.
(2) **How to determine the number of persons with the same name?** There is no answer for question like how many people having the same name with you. However, this is usually a pre-specified parameter for existing name disambiguation algorithms (using clustering).
(3) **How to integrate data continuously?** To ensure user experience, we need to minimize the delay between the arrival of a document and the time it displays in its authors' profiles and maintain the consistency after each update.
(4) **How to involve human efforts in the loop?** A completely automatic disambiguation system without any human interactions is far from sufficient. It is necessary to involve human efforts in the loop to achieve high disambiguation accuracy.

To this end, in AMiner, we design a unified framework to address the above challenges. For quantifying the similarity, we propose a global metric and local linkage learning algorithm, which projects each entity into a low-dimensional latent common space. This offers a way to directly compute the similarity of entities from different sources. For determining the number of persons who share the same name, we propose an end-to-end model that directly estimates the number of persons (clusters) in a dataset using a recurrent neural network. For involving human into disambiguation, we formally define six potential feedbacks from users/annotators. The feedbacks are then incorporated into different components of the framework to improve the disambiguation accuracy.

We evaluate the proposed framework on real-world large data. Our experiments show that the proposed solution achieves significantly better performance than several state-of-the-art methods including GHOST [33], Zhang et al. [33], and Louppe et al. [17] (+7-35% in terms of F1-score). The automatically estimated number of persons by our proposed model is close to the actual number. The framework has been deployed in AMiner to deal with the disambiguation problem at the billion scale.

2 RELATED WORK

Name disambiguation is usually viewed as a clustering problem [1, 9, 10, 12, 22, 32]. As many other clustering tasks, there are two main challenges that need to be addressed how to quantify the similarity and how to determine cluster size. Most existing literature mainly focuses on addressing the first sub-problem, while ignore the second. The state-of-the-art solutions for name disambiguation problem can be divided into two categories: feature-based and linkage-based.

Feature-based methods. Feature-based methods [8, 10, 22, 32] leverage supervised learning method to learn a pairwise distance function between documents based on their feature vectors. Huang et al. [10] first use blocking [22] technique to group candidate documents with similar names together. Then it learns distance between documents by an efficient Support Vector Machines (SVM) and finally employs DBSCAN to cluster documents. Yoshida et al. [32] propose a two-stage clustering method to learn better feature representation via the first clustering step. Han et al. [8] present supervised disambiguation methods based on SVM and Naïve Bayes. Moreover, Louppe et al. [17] use a classifier to learn pairwise similarity and perform semi-supervised hierarchical clustering to generate results.

Linkage-based methods. There are also several linkage/graph-based methods [1, 9, 12] to deal with name disambiguation. These methods are capable of utilizing graph topology and aggregate information from neighbors. GHOST [5] builds document graph for each ambiguous name by co-authorship only. It uses carefully-designed similarity function and uses affinity propagation algorithm to generate clustering results. Tang et al. [23] employ Hidden Markov Random Fields to model node features and edge features in a unified probabilistic framework. Zhang et al. [33] solve this problem by learning graph embedding from three constructed graphs based on document similarity and coauthor relationship.

Our method proposed in this paper combines the advantages of above two methods by learning a global embedding using supervised metric learning and refining the embedding using local linkage structures.

Cluster size estimation. The second challenge is how to determine the number K of clusters (name identities). Most previous literature assumes the number is known beforehand and ignores this problem in the solution. Several existing works claim to use clustering methods such as DBSCAN to avoid specifying K; however several density-based hyperparameters are still needed to be pre-specified. Tang et al. [23] use a variation of X-means [19] algorithm to iteratively estimates the optimal K by measuring clustering quality based on Bayesian Information Criterion (BIC). However, in our empirical study on large data, we found that the BIC-based methods are inclined to merge clusters together, thus result in low accuracy. In this paper, we seek to learn an end-to-end model that takes a set of document embeddings as input and directly predict the number of clusters.

There is also a thread of research [21, 29] solving this problem in a hierarchical tree model instead of making pairwise comparisons

using clustering. For example, Wick et al. [29] build a discriminative hierarchical factor graph model for coreference, and the hierarchical model can also address the scalability issue. Furthermore, human edits can be incorporated [28] by running MCMC inference on the hierarchical model.

3 PRELIMINARIES

In this section, we present the formulation of the problem with preliminaries.

3.1 Problem Formulation

Let a be a given name reference, and $\mathcal{D}^a = \{D_1^a, D_2^a, ..., D_N^a\}$ be a set of N documents associated with the author name a. We call $\mathcal{D}^a$ as the **candidate set** of a. Each document $D_i^a \in \mathcal{D}^a$ is represented by a set of features $D_i^a = \{x_1, x_2, ...\}$ including title, abstracts, co-author names, venue names, etc. We use $\mathbb{I}(D_i^a)$ to denote the identity (corresponding real-world person) of D_i^a. Thus if D_i^a and D_j^a are authored by the same author, we have $\mathbb{I}(D_i^a) = \mathbb{I}(D_j^a)$. Given this, we define the problem of author disambiguation as follows.

Definition 3.1. Name Disambiguation. The task of author disambiguation is to find a function Φ to partition $\mathcal{D}^a$ into a set of disjoint clusters, i.e.,

$$\Phi(\mathcal{D}^a) \rightarrow \mathbf{C}^a, \text{where } \mathbf{C}^a = \{C_1^a, C_2^a, ..., C_K^a\},$$

such that each cluster only contains documents of the same identity—i.e., $\mathbb{I}(D_i^a) = \mathbb{I}(D_j^a), \forall (D_i^a, D_j^a) \in C_k^a \times C_k^a$, and different clusters contains documents of different identities—i.e., $\mathbb{I}(D_i^a) \neq \mathbb{I}(D_j^a), \forall (D_i^a, D_j^a) \in C_k^a \times C_{k'}^a, k \neq k'$.

We call $\mathbf{C}^a$ a disambiguation (clustering) solution of $\mathcal{D}^a$ for name a. We omit the superscript a in the following description if there is no ambiguity. In general, the problem is very difficult, if there is not any supervision information. Human annotators (or users) can provide useful constraints. We mainly consider two kinds of constraints: identity constraints S^I and pairwise constraints S^P, with $S = S^I \cup S^P$. The identity constraint $(D_i, C_k, y_{ik} \in \{0, 1\})$ indicates that D_i should belong to (or not belong to) the cluster C_k

$$\begin{cases} (D_i, C_k, 0) \in S^I \rightarrow D_i \notin C_k, \\ (D_i, C_k, 1) \in S^I \rightarrow D_i \in C_k. \end{cases}$$

The pairwise constraint $(D_i, D_j, y_{ij} \in \{0, 1\})$ indicates that the two documents D_i and D_j should (or not) belong to the same author.

$$\begin{cases} (D_i, D_j, 0) \in S^P \rightarrow \mathbb{I}(D_i) \neq \mathbb{I}(D_j), \\ (D_i, D_j, 1) \in S^P \rightarrow \mathbb{I}(D_i) = \mathbb{I}(D_j). \end{cases}$$

Please note that identity constraints imply pairwise constraints since

$$\begin{cases} (D_i, C_k, 1) \in S^I \wedge (D_j, C_k, 1) \in S^I \rightarrow (D_i, D_j, 1) \in S^P, \\ (D_i, C_k, 1) \in S^I \wedge (D_j, C_{k'}, 1) \in S^I \wedge C_k \neq C_{k'} \rightarrow (D_i, D_j, 0) \in S^P, \\ (D_i, C_k, 1) \in S^I \wedge (D_j, C_k, 0) \in S^I \rightarrow (D_i, D_j, 0) \in S^P. \end{cases} \quad (1)$$

For simplicity, we assume there is no logical conflict in S.

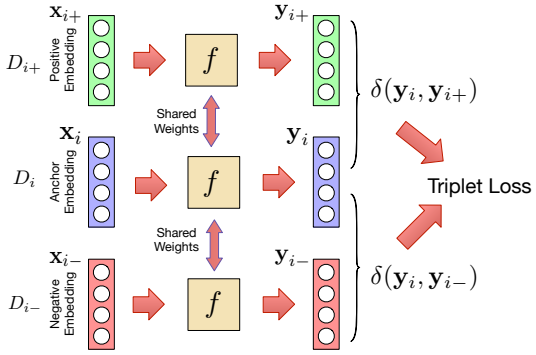


Figure 2: The architecture of global metric learning with triplet loss. Each training example is a triplet consisting of an anchor D_i, a positive sample D_{i+} and a negative sample D_{i-}. The objective is to distinguish positive pairs (D_i, D_{i+}) from negative pairs (D_i, D_{i+}).

3.2 AMiner

AMiner is a free online academic search and mining system and also the second generation of ArnetMiner [25], with the emphasis to offer approaches to gain a deeper understanding of the large and heterogeneous information networks (authors, papers, venues, topics, etc.) formed in the scientific literature data. The system has been in operation since 2006 and has attracted more than 8,000,000 independent IP accesses from over 220 countries/regions. It has become strategic partner of Microsoft Academic Search and the official content provider of Sogou[3] Scholar.

AMiner automatically extracts researchers' profiles from the Web and integrates the publication data from online databases such as DBLP, ACM Digital Library, CiteSeer, and SCI. In the integration, we inevitably have to face the challenge of name ambiguity. The problem is becoming more and more challenging as the volume of publication data is growing at a rapid rate, with more than 500,000 new documents needed to be integrated into the system per month. In this paper, we systematically discuss how to solve this problem at the billion scale and in an incremental fashion.

4 FRAMEWORK

In this section, we discuss the design and implementation of our deployed solution to author disambiguation problem in detail. We first propose our representation learning method and introduce how to estimate cluster size in the disambiguation process. To deploy the algorithm in a large scale online system, we need to minimize the delay of the integration and maintain the consistency after each update. We discuss our solution for continuous integration in Section 4.3 and present how to leverage human annotations in Section 4.4.

4.1 Representation Learning

In order to effectively quantify the similarity between different documents, we first transform documents into an embedding space such that D_i is close to D_j if $\mathbb{I}(D_i) = \mathbb{I}(D_j)$. There are two ways to embed the data. We can either learn a global embedding function

[3] http://sogou.com, the second largest search engine in China.

that encodes all the documents in a unified space or learn a local embedding function for each candidate set separately. In our framework, we first learn a supervised global embedding function, then refine the global embeddings for each candidate set based on the local contexts. We first present the implementation of our method and then discuss the intuition of our design in Section 4.5.

4.1.1 Global Metric Learning. Input document D_i is represented as a varied-length set of features $D_i = \{x_1, x_2, ...\}$, where features are words in title and abstract, coauthor names, venue names, affiliations, etc. Each feature is a one-hot vector. We first transform the feature set into a continuous low-dimensional space. Inspired by the unsupervised representation learning techniques, we use Word2Vec [18] to obtain an embedding $\bar{x}_n \in \mathbb{R}^d$ for each feature x_n. We define the feature embedding of document D_i as $\mathbf{x}_i = \sum_{x_n \in D_i} \alpha_n \bar{x}_n$, which is a weighted sum of the embedding of each feature in D_i. In the equation, α_n is the inverted document frequency of feature x_n and $\mathbf{x}_i$ captures the correlations between features based on the co-occurrence statistics within each individual documents [18]. However, the ability of $\mathbf{x}_i$ to distinguish documents with different authors is limited. Thus, we seek to leverage labeled data to fine-tune the embedding.

Contrastive Loss. Given a set of constraints $S^P = \{(D_i, D_j, y)\}$ where $y = 1$ if $\mathbb{I}(D_i) = \mathbb{I}(D_j)$ (i.e. positive pairs) and $\mathbb{I}(D_i) \neq \mathbb{I}(D_j)$ if $y = 0$ (i.e. negative pairs). The idea of metric learning is to enforce positive pairs to be close in the embedding space and negative pairs to be far away. To this end, we introduce another embedding function $f : \mathbf{x}_i \in \mathbb{R}^d \rightarrow \mathbb{R}^{d_f}$, and $\mathbf{y}_i = f(\mathbf{x}_i)$ as a new embedding of D_i. A straightforward idea is to optimize the contrastive loss $\mathcal{L}_f = \sum_{(D_i, D_j, y) \in S^P} \mathcal{L}_f(D_i, D_j, y)$ where

$$\mathcal{L}_f(D_i, D_j, y) = \begin{cases} \delta(\mathbf{y}_i, \mathbf{y}_j), & y = 1 \\ \max\{0, m - \delta(\mathbf{y}_i, \mathbf{y}_j)\}, & y = 0 \end{cases}$$

where $\delta(v_1, v_2) = ||v_1 - v_2||_2$ is the euclidean distance in the embedding space, $m > 0$ is a margin. $\mathcal{L}_f$ encourages all documents of the same author to be projected onto a single point in the embedding space which can be potentially troublesome since a single author might work on different topics and collaborate with different communities. Thus, we adopt Learning to Rank and optimize a triplet loss function.

Triplet Loss. Let (D_i, D_{i+}, D_{i-}) be a triplet where $\mathbb{I}(D_i) = \mathbb{I}(D_{i+})$ and $\mathbb{I}(D_i) \neq \mathbb{I}(D_{i-})$, we have

$$\begin{aligned} \delta(\mathbf{y}_i, \mathbf{y}_{i+}) + m < \delta(\mathbf{y}_i, \mathbf{y}_{i-}), \\ \forall (D_i, D_{i+}, D_{i-}) \in \mathcal{T}, \end{aligned} \tag{2}$$

where $\mathcal{T}$ is the set of all possible triplets in the training set, m is a margin enforced between positive pairs and negative pairs. The objective $\mathcal{L}_f$ is then replaced as

$$\mathcal{L}_f = \sum_{(D_i, D_{i+}, D_{i-}) \in \mathcal{T}} \max\{0, \delta(\mathbf{y}_i, \mathbf{y}_{i+}) - \delta(\mathbf{y}_i, \mathbf{y}_{i-}) + m\}. \tag{3}$$

Instead of projecting to a single point, triplet loss enables documents with the same identity to reside on a manifold [20], and at the same time maintain a distance from other documents. The architecture of global metric learning with triplet loss is shown in Figure 2.

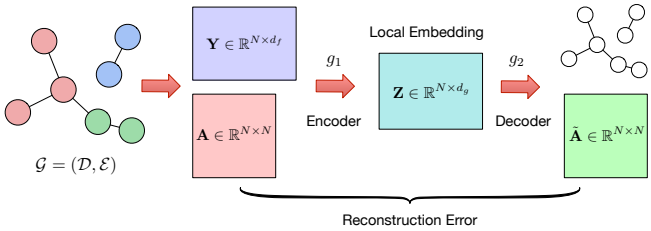


Figure 3: The architecture of local linkage learning with graph auto-encoder. $\mathcal{G}$ is a local linkage graph, Y is a feature matrix, A is an adjacency matrix, Z is a learned latent local embedding matrix, $\tilde{\mathbf{A}}$ is a reconstructed adjacency matrix. The objective is to minimize the reconstruction error between A and $\tilde{\mathbf{A}}$.

We call $\{\mathbf{y}_i\}$ global embeddings because documents in different candidate sets are embedded in a unified space. However, since the clustering is conducted separately for each name reference, local contexts within a candidate set can be leveraged to further improve the performance.

4.1.2 Local Linkage Learning. We seek to refine the global embeddings by leveraging fine-grained information within a candidate set. Since most candidate sets do not have any labeled data, an unsupervised method is preferred in this step. Following the idea of linkage based disambiguation methods proposed in [1, 9, 12, 33], we construct a graph for each candidate set.

Definition 4.1. Local Linkage Graph. For a given name reference a, we construct a local linkage graph $\mathcal{G}^a = (\mathcal{D}^a, \mathcal{E}^a)$, where $\mathcal{D}^a = \{D_i^a\}$ is the set of documents authored by a person named a, $\mathcal{E}^a = \{(D_i^a, D_j^a)\}$ is a set of edges capturing the similarity between the documents.

We measure the similarity of two documents based on the common features shared by the two documents. Let the common feature set of D_i and D_j be the intersection between their feature sets $D_i \cap D_j$, we define the linkage weight $W(D_i, D_j) = \sum_{x \in D_i \cap D_j} w_x$ between D_i and D_j as a weighted sum of all the common features, where w_x is the weight of feature x. The weights $\{w_x\}$ can be learned by a supervised model, but for simplicity, we define feature weight as its inverted document frequency. We construct an edge in $\mathcal{G}$ between D_i and D_j if $W(D_i, D_j)$ is above a threshold[4].

The intuition for constructing $\mathcal{G}$ is that D_i and D_j are likely to have the same identity if they share a lot of unique features (i.e. $W(D_i, D_j)$ is large). Thus, we try to leverage the structure of $\mathcal{G}$ to improve the global embedding.We use an unsupervised auto-encoder architecture [14, 26] to learn from the local linkage graph.

Graph Auto-encoder. Let $\mathbf{Y} = [\mathbf{y}_1^\intercal, ..., \mathbf{y}_N^\intercal]^\intercal$ denotes an embedding matrix of $\mathcal{D}$ generated by f, $\mathbf{A} \in \mathbb{R}^{N \times N}$ is the adjacency matrix of $\mathcal{G}$. A graph auto-encoder is comprised of a node encoder model $\mathbf{Z} = g_1(\mathbf{Y}, \mathbf{A})$ and an edge decoder model $\tilde{\mathbf{A}} = g_2(\mathbf{Z})$, where $\mathbf{Z} = [z_1^\intercal, ..., z_N^\intercal]^\intercal$ is a node embedding matrix, $\tilde{\mathbf{A}}$ is a predicted

[4] We empirically set the threshold as 10.

adjacency matrix. The objective is to minimize the reconstruction error between the predicted $\tilde{\mathbf{A}}$ and the original adjacency matrix $\mathbf{A}$.

We instantiate g_1 as a two-layer graph convolution network (GCN) [13] due to its effectiveness for modeling networked data:

$$g_1(\mathbf{Y}, \mathbf{A}) = \dot{\mathbf{A}}\text{ReLU}(\dot{\mathbf{A}}\mathbf{Y}\mathbf{W}_0)\mathbf{W}_1, \tag{4}$$

where $\dot{\mathbf{A}}$ is symmetrically normalized adjacency matrix (i.e. $\dot{\mathbf{A}} = \mathbf{D}^{\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$, $\mathbf{D}$ is the degree matrix of $\mathcal{G}$), $\text{ReLU}(\cdot) = \max(0, \cdot)$, $\mathbf{W}_0$ and $\mathbf{W}_1$ are the parameters of the first and second layer respectively.

The decoder g_2 is defined as

$$g_2(\mathbf{Z}) = \text{sigmoid}(\mathbf{Z}^\intercal\mathbf{Z}) \tag{5}$$

Thus, the probability of predicting the existence of an edge between D_i and D_j is given by

$$p(\tilde{\mathbf{A}}_{ij} = 1|\mathbf{z}_i, \mathbf{z}_j) = \text{sigmoid}(\mathbf{z}_i^\intercal\mathbf{z}_j) \tag{6}$$

The objective is then defined as minimizing the cross entropy:

$$\mathcal{L}_g = -\sum_{D_i, D_j \in \mathcal{D}} \mathbf{A}_{ij} \log p(\tilde{\mathbf{A}}_{ij}) \tag{7}$$

We take the latent variables $\mathbf{Z} = [\mathbf{z}_1^\intercal, ..., \mathbf{z}_N^\intercal]^\intercal$ as new document embeddings. $\mathbf{Z}$ incorporates information from both global and local context. In our implementation, we use a variational version of graph auto-encoder [14] by assuming $\mathbf{Z}$ is generated from a latent Gaussian distribution, hence, Equation (7) is extended as the sum of a reconstruction loss and the KL divergence between the learned latent distribution and the prior distribution. Our empirical results show that variational graph auto-encoder outperforms the original one. The binary adjacency matrix $\mathbf{A}$ in the encoder model can be replaced with the weight matrix calculated by $W(D_i, D_j)$. However, in our experiments, we found that it leads to slower convergence and does not significantly improve the performance.

The local linkage learning model is unsupervised since we construct the local linkage graph $\mathcal{G}$ purely based on the overlap of features between documents. We can refine $\mathcal{G}$ based on the pairwise constraints S^P to support semi-supervised learning (Cf. § 4.4).

4.2 Cluster Size Estimation

Based on the learned embeddings, the final partition of a candidate set is determined by a clustering algorithm. We use hierarchical agglomerative clustering algorithm (HAC) as our main clustering method. One common issue for clustering problems is how to estimate the number of clusters K. There are existing clustering methods such as DBSCAN that do not take an explicit K as input, but it requires preset hyperparameters to determine density which changes significantly in different candidate sets. The most popular way to determine the number of clusters is X-means [19], which iteratively splitting the centroids and searching for an optimal K based on the quality of the proposed clustering. There are mainly two drawbacks of X-means algorithm for our task. First, the clustering quality is scored based on a predefined measurement such as Bayesian Information Criterion which cannot handle a complex mixture of data (hence tend to over merge the data when the number of clusters is large) in high dimension [7]. Second, it involves an iterative trial of potential splits which is not efficient enough on large datasets. To overcome the above issues, we seek to learn an end-to-end model $h(\mathcal{D}) \rightarrow \mathbb{R}$ that takes a set of documents as input and directly estimates the number of different identities in the candidate set.

Recent progress in deep learning community shows the capacity of recurrent neural networks in modeling discrete sequences and sets of data. Bello et al. [2] shows that RNN is able to act as an encoder in the solution of combinatorial optimization problems such as TSP. Inspired by these works, we adopt RNN as an encoder and try to map a set of embedding vectors to the true number of clusters in the set. To achieve this goal, there are two challenges. First, the size of a candidate set varies in a wide range from one to tens of thousands. Although RNN is able to handle variable sized input via padding and truncation, those operations may introduce unwanted bias. Second, it is hard to construct a training set for this problem. It is infeasible to manually label a large number of candidate sets with true cluster size. To resolve these issues, we propose a sampling strategy to construct a pseudo-training set.

Let $\mathbf{C} = \{C_1, C_2, ...\}$ be a set of clean clusters (as discussed in Section 4.4) where each cluster only contains documents of a single author. The clusters may be from different candidate sets. For each t^{th} training step, we first uniformly sample the number of clusters K_t from $[K_{\min}, K_{\max}]$. Then, we sample K_t clusters from $\mathbf{C}$ to construct a pseudo-candidate set $\mathbf{C}_t$. Let $\mathcal{D}_{\mathbf{C}_t} = \bigcup_{C_i \in \mathbf{C}_t}\{D \in C_i\}$ denotes all the documents within $\mathbf{C}$ and z denotes a fixed number of sample size. We then sample a set of z documents $\mathcal{D}_t$ from $\mathcal{D}_{\mathbf{C}_t}$ with replacement. We note that $\mathcal{D}_t$ may contain duplicate documents and the order of $\mathcal{D}_t$ is arbitrary. In this way, we can construct infinite amount of pseudo-training examples from $\mathbf{C}$. We use a neural network architecture $h(\mathcal{D}_t) \rightarrow \mathbb{R}$ with a bi-directional LSTM as encoder and a one-dimensional fully-connected layer as decoder. The model takes the raw feature embedding $\mathbf{x}_i$ of each document $D_i \in \mathcal{D}_t$ as input. We optimize the Mean Squared Logarithmic Error as follows:

$$\mathcal{L}_h = \frac{1}{N}\sum_{t=1}^{a}[\log(1 + h(\mathcal{D}_t)) - \log(1 + K_t)]^2 \tag{8}$$

Algorithm 1 summarizes the pseudo-training data generation strategy for cluster size estimation described above. Our experimental results (in Section 5.6) suggest that in our task the proposed method is significantly better than traditional X-means approach.

4.3 Continuous Integration

Our system integrates new scholarly documents in a streaming fashion. Based on our statistics, there are more than 500,000 new documents being integrated into the system every month. Thus, it is crucial for us the efficiently handle the ever-growing data volume. Although efficiency is an important objective for designing our disambiguation method, it is still impossible for us to re-compute the clustering from scratch for every single new document. The main time cost comes from local linkage learning, clustering and the IO overhead of retrieving all the documents associated with a candidate set from the database. Thus, instead, we maintain a

ALGORITHM 1: Pseudo-training data generation strategy for cluster size estimation. $(\mathcal{D}_t, K_t)$ is a training example for RNN model $h(\mathcal{D}) \rightarrow \mathbb{R}$.

Input: Clean clusters $\mathbf{C}$, $K_{\min}$, $K_{\max}$, sample size z, step t;
Output: Pseudo-training example $(\mathcal{D}_t, K_t)$;
$K_t \leftarrow$ Sample from $[K_{\min}, K_{\max}]$;
$\mathbf{C}_t \leftarrow$ Sample K_t clusters from $\mathbf{C}$;
$\mathcal{D}_t \leftarrow$ Sample z documents from $\bigcup_{C_i \in \mathbf{C}_t} \{D \in C_i\}$ with replacement;
return $(\mathcal{D}_t, K_t)$;

priority queue of every candidate sets and update each candidate set in an iterative manner. However, a full update of all the candidate sets usually takes weeks which is not acceptable for an online system.

Real-time Update. To overcome this issue, each new document D^* will first be greedily assigned to an existing profile in the following way. We search for a set of profiles $\text{Hits}(D^*) = \{C_k\}$ based on the author name and affiliation using an inverted index of all the profiles in the system where each profile is corresponding to a cluster of documents. If there are multiple hits, we retrieve the global embeddings $\{\mathbf{y}_i\}$ of documents $\{D_i \in \bigcup C_k\}$ and construct a local kNN classifier to find the best assignment where each C_k is a class and $\{(\mathbf{y}_i, C_{\hat{k}}) | D_i \in C_{\hat{k}}\}$ is a set of data points (documents) with their labels (corresponding profiles). This simple strategy enables us to update the documents almost in real-time. Although the assignment may be sub-optimal, it will be corrected by the next iteration of clustering re-computation.

Data Consistency. Another issue we are facing with in an online system is how to maintain the consistency between each iteration of the updates. Due to the unsupervised nature of clustering, the index of each cluster is anonymous. Thus, after re-computing the clustering, the index of clusters might be inconsistent with the former ones. This is critical for an online system as each cluster is associated with an individual profile, herein a change may significantly harm user experience. Hence, after obtaining a new clustering $\mathbf{C}_{t+1}$, we search for an optimal matching between it and its previous version $\mathbf{C}_t$. We define the optimal matching $\hat{M}_{\mathbf{C}_t \mathbf{C}_{t+1}}$ as:

$$\hat{M}_{\mathbf{C}_t \mathbf{C}_{t+1}} = \arg \max_{M_{\mathbf{C}_t \mathbf{C}_{t+1}}} \sum_{(C_i, C_j) \in M_{\mathbf{C}_t \mathbf{C}_{t+1}}} \text{sim}(C_i, C_j), \tag{9}$$

subject to one-to-one matching constraint, where $\text{sim}(C_i, C_j)$ is defined as the Jaccard coefficient. An optimal one-to-one mapping is found using Kuhn-Munkres algorithm [15].

4.4 Human in the Loop

The automatically generated clusterings are far from being perfect. Thus, to improve the accuracy we need to involve human in the loop of author disambiguation. In our system, we allow both users and professional annotators to provide feedback based on the clustering results. The supported feedbacks include:

(1) **Delete:** removing a document D_i from a profile C_k and adding an identity constraint $(D_i, C_k, 0)$ into S^{I}.
(2) **Insert:** adding a document D_i in to a profile C_k and adding an identity constrain $(D_i, C_k, 1)$ into S^{I}.
(3) **Split:** annotating a profile C_k as over-merged and requesting clustering within the profile.
(4) **Merge:** merge all the documents from C_k to $C_{k'}$.
(5) **Create:** creating an empty profile C_k.
(6) **Confirm:** label a profile C_k as a clean cluster and adding $\{(D_i, C_k, 1) | D_i \in C_k\}$ into S^{I}.

To leverage user feedbacks in the algorithm, we translate the identity constraints S^{I} to pairwise constraints S^{P} according to Equation (1). Pairwise constraints are used in both of the two embedding learning phases.

In global metric learning phrase (Section 4.1.1), the training set $\mathcal{T}$ is generated from S^{P} in the following way: 1) Sample a constraint (D_i, D_j, y_{ij}) from S^{P}. 2) If $y_{ij} = 0$, sample a constraint $(D_i, D_l, 1)$ from S^{P} and generate a triplet (D_i, D_l, D_j). 3) Otherwise, sample a random document $D_{l'}$ from the whole document space and generate a triplet $(D_i, D_j, D_{l'})$.

In local linkage learning phrase (Section 4.1.2), we refine the local linkage graph $\mathcal{G} = (\mathcal{D}, \mathcal{E})$ based on S^{P}. We add an edge (D_i, D_j) into $\mathcal{E}$ if

$$(D_i, D_j, 1) \in S^{\text{P}} \wedge (D_i, D_j) \notin \mathcal{E} \wedge (D_i, D_j) \in \mathcal{D} \times \mathcal{D},$$

and remove edge (D_i, D_j) from $\mathcal{E}$ if

$$(D_i, D_j, 0) \in S^{\text{P}} \wedge (D_i, D_j) \in \mathcal{E}.$$

Identity constraints S^{I} are also used to generate pseudo-training set for clustering size estimation described in Section 4.2. The clean cluster set $\mathbf{C}$ is constructed by:

$$\mathbf{C} = \{C_k | \exists (\cdot, C_k, 1) \in S^{\text{I}}\}, \text{where } C_k = \{D_i | (D_i, C_k, 1) \in S^{\text{I}}\}$$

4.5 Discussion

The global metric learning model f is *inductive* in the sense that it is able to be employed by a new document that has not been seen in the training time, while the local linkage model g is *transductive*, meaning that it is not able to generalize to unseen instances.

Empirically, with limited labeled data, the local model performs better since it leverages fine-grain information within a candidate set. However, the global model is particularly useful in our system for two reasons. First, most candidate sets do not have any labeled data since the human annotations are concentrated on a small fraction of the profiles. The global model is able to transfer external supervision from other candidate sets to improve the performance of the local model. Second, as we described in Section 4.3, constructing and learning from the local linkage graph is time-consuming. It is infeasible for us to re-train the local model for every single new document. The inductive nature of the global model enables us to instantly determine a fairly good temporary assignment for an incoming document.

5 EXPERIMENTS

All codes and data used in this work are publicly available.[5]

[5] https://github.com/neozhangthe1/disambiguation/

Table 1: Results of Author Name Disambiguation.

	AMiner			Zhang et al.			GHOST			Louppe et al.			Rule		
Name	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1
Xu Xu	74.18	45.86	56.68	48.16	41.87	44.80	61.34	21.79	32.15	22.55	64.40	33.40	10.75	97.23	19.35
Rong Yu	89.13	46.51	61.12	65.48	40.85	50.32	92.00	36.41	52.17	38.85	91.43	54.53	30.81	97.79	46.86
Yong Tian	76.32	51.95	61.82	70.74	56.85	63.04	86.94	54.58	67.06	32.08	63.71	42.67	10.37	93.79	18.67
Lu Han	51.78	28.05	36.39	47.88	20.62	28.82	69.72	17.39	27.84	30.25	46.65	36.70	13.66	89.16	23.69
Lin Huang	77.10	32.87	46.09	71.84	34.17	46.31	86.15	17.25	28.74	24.86	71.32	36.87	13.86	99.46	24.33
Kexin Xu	91.37	98.64	94.87	90.02	82.47	86.08	92.90	28.52	43.64	91.26	98.35	94.67	91.45	99.60	95.35
Wei Quan	53.88	39.02	45.26	64.45	47.66	54.77	86.42	27.80	42.07	37.86	63.41	47.41	28.16	93.80	43.32
Tao Deng	81.63	43.62	56.86	53.04	29.89	38.23	73.33	24.50	36.73	40.46	51.38	45.27	16.30	95.16	27.84
Hongbin Li	77.20	69.21	72.99	54.66	53.05	53.84	56.29	29.12	38.39	19.48	85.96	31.77	13.25	96.41	23.30
Hua Bai	71.49	39.73	51.08	58.58	35.90	44.52	83.06	29.54	43.58	36.39	41.33	38.70	25.47	98.51	40.47
Meiling Chen	74.93	44.70	55.99	59.36	28.80	38.79	86.11	23.85	37.35	58.32	47.14	52.14	59.55	82.07	69.02
Yanqing Wang	71.52	75.33	73.37	60.40	51.97	55.87	80.79	40.39	53.86	29.64	79.08	43.11	25.72	62.47	36.44
Xudong Zhang	62.40	22.54	33.12	70.20	23.35	35.04	85.75	7.23	13.34	72.38	79.83	75.92	63.22	17.94	27.95
Qiang Shi	52.20	36.15	42.72	43.84	36.94	40.10	53.72	26.80	35.76	35.31	47.18	40.39	28.79	93.89	44.06
Min Zheng	57.65	22.35	32.21	54.76	19.70	28.98	80.50	15.21	25.58	25.86	32.67	28.87	15.41	98.72	26.66
Avg.	77.96	63.03	**67.79**	70.63	59.53	62.81	81.62	40.43	50.23	57.09	77.22	63.10	44.94	89.30	53.42

5.1 Data Summarization

To systematically evaluate the proposed method, we construct a benchmark based on AMiner. We sampled 100 author names from a well-labeled subset of AMiner database. The benchmark consists of 70,258 documents from 12,798 authors. The labeling process was carried out based on the publication lists on the authors' homepages and the affiliations, e-mail addresses in the web databases (e.g. Scopus, ACM Digital Library, etc). We applied "majority voting" to cope with the disagreements in the annotation process. Comparing to existing benchmarks for name disambiguation[6,7], our benchmark is significantly larger (in terms of the number of documents) and more challenging (since each candidate set contains much more clusters).

5.2 Baseline Comparison Methods

To validate the performance of our proposed approach, we compare it against three state-of-the-art name disambiguation methods. For a fair comparison, the number of clusters is set to the true value.

Zhang et al. [33]: This method constructs three local graphs for a candidate set based on coauthors and document similarity. A graph embedding is learned for each candidate set by sampling triplets from the graph. The final result is generated by agglomerative hierarchical clustering.

GHOST [5]: The second method is purely based on coauthor names. For each query name, it constructs a graph by collapsing all the coauthors with identical names to one single node. The distance between two nodes is measured based on the number of valid paths. The final clustering result is generated by affinity propagation algorithm.

Louppe et al. [17]: This method first trains a pairwise distance function base on a set of carefully designed similarity features. A semi-supervised HAC algorithm is used to determine clusters.

[6]https://aminer.org/disambiguation
[7]http://clgiles.ist.psu.edu/data/

Rule: Rule-based method constructs local linkage graphs by connecting two documents when their co-authors, affiliations or publication venues are strictly matched. The clustering is obtained by simply partitioning the graph into connected components.

Our method is indicated by **AMiner**. In order to analyze the contribution of each component in our solution, we also present our performance at different stages in Table 2.

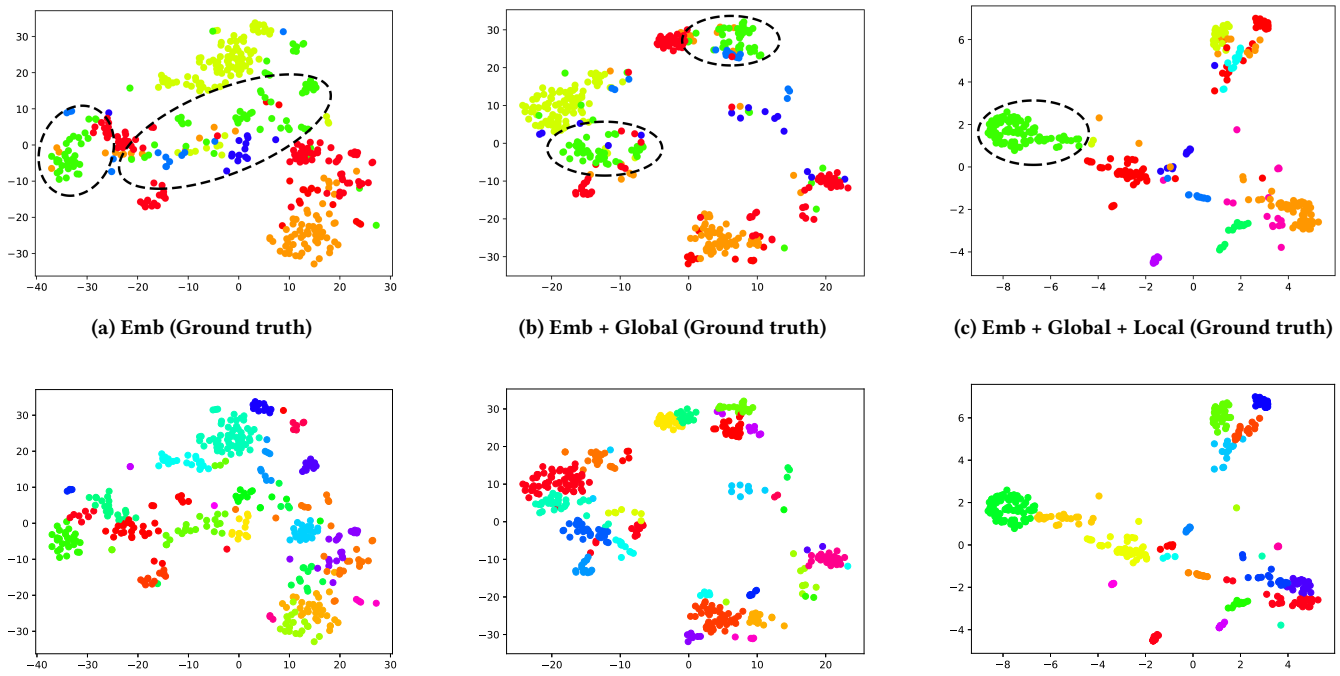
Embedding: This is the clustering result based on the original feature space, where each document D_i is represented as its feature embedding $\mathbf{x}_i$ defined in Section 4.1.1.

Global: This shows the result after global metric learning (Section 4.1.1). The embedding space is fine-tuned by optimizing a triplet loss. We use the global embeddings $\{\mathbf{y}_i\}$ as the of clustering.

Local: This method uses orthogonal one-hot vectors as the input node features of local linkage learning (Section 4.1.2).

5.3 Experimental Results

Table 1 shows the performance of different disambiguation methods on some sampled names. We use pairwise Precision, Recall, and F1-score to evaluate our method against the alternative ones. A macro averaged score of each metric is calculated according to all test names. A holdout sample of the dataset is used for training. Louppe et al. use carefully designed features to learn a supervised pairwise similarity function, instead, we avoid hand-crafted features by learning from raw input features. Both GHOST and Zhang et al. leverage the structure of coauthor graphs. GHOST directly partitions the coauthor graph by affinity propagation. Zhang et al. transforms coauthor graphs into local embeddings by sampling edges from the graph, which is closely related to our local linkage learning method. However, by incorporating both global supervision and local linkage structure, our method (AMiner) outperforms the baselines in terms of F1-score (+7.93% over Zhang et al., +34.96% over GHOST and +7.43% over Louppe et al. relatively). We also tested a baseline method that directly partitions local linkage graphs into



(a) Emb (Ground truth) (b) Emb + Global (Ground truth) (c) Emb + Global + Local (Ground truth)

(d) Emb (F1: 35.36%) (e) Emb + Global (F1: 42.75%) (f) Emb + Global + Local (F1: 61.11%)

Figure 4: t-SNE Visualization of embedding spaces on a candidate set. Each color in (a), (b), (c) denotes an individual ground truth cluster, while each color in (d), (e), (f) denotes a predicted cluster by hierarchical agglomerative clustering. Emb indicates the original feature embedding. Global and Local represent the use of global metric learning and local linkage learning respectively. The dashed black ellipses in (a), (b), (c) circle the points of the same ground truth cluster.

Table 2: Contribution of Each Component.

	Pre.	Rec.	F1
Embedding	66.85	42.04	49.79
Global	68.40	47.42	54.56
Local	68.97	67.68	66.55
Overall	77.96	63.03	67.79

connected components which yields low precision, hence verifies the effectiveness of local linkage learning.

5.4 Contribution Analysis

In Table 2, some incremental results of our method are presented. Global outperforms Embedding by +9.58% which verifies the effectiveness of leveraging global supervision. Local achieves a much better performance than Global (+18.01% in terms of F1) which shows the advantage of the local model over the global model with limited training data. AMiner outperforms Local by +1.86% in terms of F1-score and +13.03% in terms of Precision which verifies the effectiveness of incorporating global supervision in local models. In our system, a higher precision is preferred for the sake of user experience.

5.5 Embedding Analysis

In order to further evaluate the effect of global metric learning and local linkage learning, we project the embedding generated at different stages into 2-dimensional Euclidean space which can be easily visualized. Figure 4 shows the t-SNE plot of the embedding of a candidate set where each point is a document. In the first row, the color of a point denotes the corresponding ground truth cluster, while in the second row, the color denotes the cluster predicted by hierarchical agglomerative clustering.

Figure 4a shows the original feature embedding space without global metric learning and local linkage learning. We can easily observe that points from different clusters are not well separated in Figure 4a, where there are a significant amount of overlaps between different clusters. The dashed black ellipses circle the points of a green cluster which are scattered over a wide area in the space. Figure 4b shows the embedding improved by global metric learning where the green points form two clusters in the embedding space (represented by two dashed black ellipses). Figure 4c shows the final embedding after local linkage learning. We can see that the green cluster is well separated from other points in the embedding space.

Figure 4d, 4e and 4f demonstrate the clustering results generated by K-means algorithm. We achieve an F1 of 61.11% with both global

Table 3: Results of Clustering Size Estimation.

	Actual	RNN	Regression	X-means
RMSLE	-	**0.2493**	1.6006	2.1065
Song Chen	125	101.39	173.80	10
Jian Du	87	62.89	110.21	5
Fosong Wang	4	5.71	184.75	5
J Yu	346	74.06	24.92	7
Yang Shen	157	153.77	89.52	7
Xiaobing Luo	13	11.01	143.44	3
Jian Feng	102	149.73	113.88	8
Lu Han	129	114.51	173.16	7

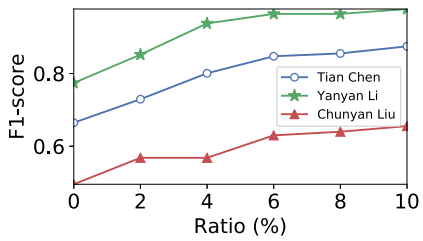


Figure 5: The performance of with pairwise constraints.

metric learning and local linkage learning which significantly outperforms the result with the original feature embedding and global metric learning only.

5.6 Cluster Size Estimation

We compare our proposed cluster size estimation method with the traditional X-means [19] algorithm with Bayesian Information Criterion as the measurement of clustering quality. We set the $K_{\min} = 1$ and $K_{\max} = 300$. Our RNN is trained according to the pseudo-training data sampling method described in Section 4.2. Half of the labeled candidate sets are used for training and the other half are used for evaluation. There is no overlap between training and test data. We evaluate the performance of both methods using Root Mean Squared Logarithmic Error. Table 3 summarizes the results. From Table 3 our method achieves a RMSLE of 0.2493 which significantly outperforms the baseline. To provide a more comprehensive view of the performance, we also demonstrate results of some individual candidate sets in Table 3 comparing to the actual number of clusters. It is easy to see that the maximum estimation from X-means is around 10 which is not usable since there can be hundreds of clusters in each candidate set. Our method provides an estimation within a reasonable error range for most cases.

To further illustrate the effectiveness of RNN encoder, we test the same method except changing the RNN to a DNN with two fully connected layers that take the summation of the feature embeddings as input. The result is labeled as Regression in Table 3. We can see that without RNN encoder, the estimation is almost random. This result verifies the ability of RNN to encode a discrete set of data points.

5.7 Effect of Constraints

In this experiment, we show the effect of adding labeled constraints into local linkage learning. As we described in Section 4.4, pairwise constraints S^P can be leveraged in local linkage learning by refining the local linkage graph $\mathcal{G}$. The experiment is conducted by sampling pairs of documents within a candidate set and adding them into the pairwise constraint set based on their ground truth clusters. Figure 5 shows the performance by labeling different amount of pairs. The X-axis is the proportion of pairs that are labeled and the Y-axis is the corresponding F1-score. We can see that the performance improves significantly with a few pairs being labeled. Active learning can be leveraged here to improve the labeling efficiency, which we intend to explore as future work.

6 CONCLUSION

To conclude, in this paper, we propose a novel representation learning framework by leveraging both global supervision and local contexts. Experimental results verify the advantage of our method over state-of-the-art name disambiguation methods. We address the problem of estimating cluster size by learning a recurrent neural network on sampled pseudo-training set. We discuss the lessons learned from deploying the name disambiguation framework in a large-scale online system and present how to leverage human annotations to improve disambiguation accuracy.

ACKNOWLEDGMENTS

Jie Tang is the corresponding author of this paper. The work is supported by the National High-tech R&D Program (2015AA124102), Development Program of China (2016QY01W0200), National Basic Research Program of China (2014CB340506), National Natural Science Foundation of China (61631013,61561130160), National Social Science Foundation of China (13&ZD190), a research fund supported by MSRA, and the Royal Society-Newton Advanced Fellowship Award.

REFERENCES

[1] Ron Bekkerman and Andrew McCallum. 2005. Disambiguating Web Appearances of People in a Social Network. In *WWW'05*. 463–470.
[2] Irwan Bello, Hieu Pham, Quoc V. Le, Mohammad Norouzi, and Samy Bengio. 2016. Neural Combinatorial Optimization with Reinforcement Learning. *CoRR* abs/1611.09940 (2016).
[3] Omar Benjelloun, Hector Garcia-Molina, David Menestrina, Qi Su, Steven Euijong Whang, and Jennifer Widom. 2009. Swoosh: a generic approach to entity resolution. *The VLDB Journal* 18, 1 (2009), 255–276.
[4] Indrajit Bhattacharya and Lise Getoor. 2007. Collective entity resolution in relational data. *TKDE* 1, 1 (2007), 5.
[5] Xiaoming Fan, Jianyong Wang, Xu Pu, Lizhu Zhou, and Bing Lv. 2011. On graph-based name disambiguation. *JDIQ* 2, 2 (2011), 10.
[6] Luis Galárraga, Geremy Heitz, Kevin Murphy, and Fabian M Suchanek. 2014. Canonicalizing open knowledge bases. In *CIKM'14*. ACM, 1679–1688.
[7] Christophe Giraud. 2014. *Introduction to high-dimensional statistics*. Vol. 138. CRC Press.
[8] Hui Han, Lee Giles, Hongyuan Zha, Cheng Li, and Kostas Tsioutsiouliklis. 2004. Two supervised learning approaches for name disambiguation in author citations. In *JCDL'04*. 296–305.
[9] Linus Hermansson, Tommi Kerola, Fredrik Johansson, Vinay Jethava, and Devdatt Dubhashi. 2013. Entity disambiguation in anonymized graphs using graph kernels. In *CIKM'13*. 1037–1046.
[10] Jian Huang, Seyda Ertekin, and C Lee Giles. 2006. Efficient name disambiguation for large-scale databases. In *PKDD'06*. Springer, 536–544.
[11] Lili Jiang, Jianyong Wang, Ning An, Shengyuan Wang, Jian Zhan, and Lian Li. 2009. Grape: A graph-based framework for disambiguating people appearances in web search. In *ICDM'09*. 199–208.

[12] Pallika H Kanani, Andrew McCallum, and Chris Pal. 2007. Improving Author Coreference by Resource-Bounded Information Gathering from the Web.. In *IJCAI*. 429–434.

[13] Thomas N Kipf and Max Welling. 2016. Semi-Supervised Classification with Graph Convolutional Networks. (2016).

[14] Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308* (2016).

[15] Harold W Kuhn. 1955. The Hungarian method for the assignment problem. *Naval Research Logistics (NRL)* 2, 1-2 (1955), 83–97.

[16] Xin Li, Paul Morie, and Dan Roth. 2004. Identification and tracing of ambiguous names: Discriminative and generative approaches. In *AAAI'04*. 419–424.

[17] Gilles Louppe, Hussein T Al-Natsheh, Mateusz Susik, and Eamonn James Maguire. 2016. Ethnicity sensitive author disambiguation using semi-supervised learning. In *KESW'16*. 272–287.

[18] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS'13*. 3111–3119.

[19] Dan Pelleg and Andrew W Moore. 2000. X-means: Extending K-means with Efficient Estimation of the Number of Clusters. In *ICML'00*. 727–734.

[20] Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *CVPR'15*. 815–823.

[21] Sameer Singh, Amarnag Subramanya, Fernando Pereira, and Andrew McCallum. 2011. Large-scale cross-document coreference using distributed inference and hierarchical models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, 793–803.

[22] Rebecca C. Steorts, Samuel L. Ventura, Mauricio Sadinle, and Stephen E. Fienberg. 2014. A Comparison of Blocking Methods for Record Linkage. In *PSD'14*. 253–268.

[23] Jie Tang, A.C.M. Fong, Bo Wang, and Jing Zhang. 2012. A Unified Probabilistic Framework for Name Disambiguation in Digital Library. *IEEE TKDE* 24, 6 (2012), 975–987.

[24] Jie Tang, Limin Yao, Duo Zhang, and Jing Zhang. 2010. A Combination Approach to Web User Profiling. *ACM TKDD* 5, 1 (2010), 1–44.

[25] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. 2008. ArnetMiner: Extraction and Mining of Academic Social Networks. In *KDD'08*. 990–998.

[26] Fei Tian, Bin Gao, Qing Cui, Enhong Chen, and Tie-Yan Liu. 2014. Learning deep representations for graph clustering. In *AAAI'14*. 1293–1299.

[27] David Wagner. 2002. A generalized birthday problem. In *Crypto'17*. 288–304.

[28] Michael Wick, Ari Kobren, and Andrew McCallum. 2013. Probabilistic Reasoning about Human Edits in Information Integration. In *ICML Workshop: Machine Learning Meets Crowdsourcing, Atlanta*.

[29] Michael Wick, Sameer Singh, and Andrew McCallum. 2012. A discriminative hierarchical model for fast coreference at large scale. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*. Association for Computational Linguistics, 379–388.

[30] Jianmin Wu, Tea Vallenius, Kristian Ovaska, Jukka Westermarck, Tomi P Mäkelä, and Sampsa Hautaniemi. 2009. Integrated network analysis platform for protein-protein interactions. *Nature methods* 6, 1 (2009), 75–77.

[31] Xiaoxin Yin, Jiawei Han, and Philip S Yu. 2007. Object distinction: Distinguishing objects with identical names. In *ICDE'07*. 1242–1246.

[32] Minoru Yoshida, Masaki Ikeda, Shingo Ono, Issei Sato, and Hiroshi Nakagawa. 2010. Person name disambiguation by bootstrapping. In *SIGIR'10*. ACM, 10–17.

[33] Baichuan Zhang and Mohammad Al Hasan. 2017. Name disambiguation in anonymized graphs using network embedding. In *CIKM'17*. 1239–1248.

[34] Yutao Zhang, Jie Tang, Zhilin Yang, Jian Pei, and Philip S Yu. 2015. Cosnet: Connecting heterogeneous social networks with local and global consistency. In *KDD'15*. 1485–1494.