CONNA: Addressing Name Disambiguation on The Fly

Bo Chen, Jing Zhang*, Jie Tang, *Senior Member, IEEE*, Lingfan Cai,
Zhaoyu Wang, Shu Zhao, Hong Chen, Cuiping Li

Abstract— Name disambiguation is a key and also a very tough problem in many online systems such as social search and academic search. Despite considerable research, a critical issue that has not been systematically studied is *disambiguation on the fly* — to complete the disambiguation in the real-time. This is very challenging, as the disambiguation algorithm must be accurate, efficient, and error tolerance. In this paper, we propose a novel framework — CONNA — to train a matching component and a decision component jointly via reinforcement learning. The matching component is responsible for finding the top matched candidate for the given paper, and the decision component is responsible for deciding on assigning the top matched person or creating a new person. The two components are intertwined and can be bootstrapped via jointly training. Empirically, we evaluate CONNA on two name disambiguation datasets. Experimental results show that the proposed framework can achieve a 1.21%-19.84% improvement on F1-score using joint training of the matching and the decision components. The proposed CONNA has been successfully deployed on AMiner — a large online academic search system.

Index Terms—Name disambiguation, Joint model, Multi-field multi-instance

◆

1 INTRODUCTION

Name disambiguation, aiming at disambiguating who is who, is one of the fundamental problems of the online academic network platforms such as Google Scholar, Microsoft Academic and AMiner. The problem has been extensively studied for decades [9], [13], [21], [35], [40], [42], [47] and most of the works focus on how to group the papers belonging to same persons together into a cluster from scratch. However, online academic systems have already maintained a huge number of person profiles, which are made by the "from scratch" algorithms or human beings. Out of the consideration of the computation and time cost of the real systems, it is not practical to re-compute the clusters from scratch for the new arriving papers every day. We need a more effective way to deal with the problem of name disambiguation on the fly.

This paper takes AMiner as the basis to explain how we deal with the name ambiguity problem when continuously updating persons' profiles. AMiner is a free online academic search and mining system [37], which has already extracted 133,204,120 researchers' profiles from the Web [36] and integrated with 263,781,570 papers from heterogeneous publication databases [47]. Currently, the

- *Bo Chen, Jing Zhang, Lingfan Cai, Hong Chen and Cuiping Li are with Information School, Renmin University of China, Beijing, China. E-mail:{bochen, zhang-jing, clfan, chong, licuiping}@ruc.edu.cn *Corresponding author*
- *Jie Tang is with Department of Computer Science and Technology, Tsinghua University, and Tsinghua National Laboratory for Information Science and Technology (TNList), Beijing, China, 100084. E-mail:jietang@tsinghua.edu.cn*
- *Zhaoyu Wang and Shu Zhao are with School of Computer Science and Technology, Anhui University. E-mail:e17301101@stu.ahu.edu.cn, zhaoshuzs@ahu.edu.cn*
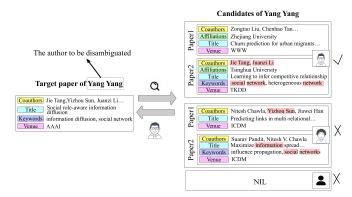

Fig. 1. Disambiguation on the fly. Given a target paper with target author as "Yang Yang", we aim at searching for the right person of "Yang Yang" from the candidates, where the right person can be a real person or a non-existing candidate denoted as NIL.

newly arrived papers of AMiner are more than 500,000 per month. How to correctly assign these papers to the right persons in the system on the fly is a critical problem for many upper applications such as expert finding, academic evaluation, reviewer recommendation and so on.

Existing methods on addressing the similar problem of anonymous author identification [2], [46] are possible solutions to continuously disambiguating papers on the fly. However, they merely target at finding the top matched person from all the candidates, but fail to deal with the situation when no right person exists, which is common in real academic systems. For example, the papers published by new researchers should not be assigned to any persons, as their profiles have not been established by the system. Thus, to assign a paper on the fly, we need to pay attention

to not only find the top matched candidate, but also identify whether to assign the top matched candidate or create a new person. In other words, we consider the absence of the right person from the candidates to be a distinct candidate, the so-called NIL candidate. Figure 1 illustrates the problem to be solved in the paper, where given a paper with an author to be disambiguated, the returned right person can be a real person or a non-existing candidate denoted as NIL. Actually, in AMiner, in addition to the "on-the-fly" assignment, we also perform a "from scratch" algorithm to cluster "NIL" papers into new profiles, and run an offline "checking" algorithm to correct errors from historical profiles periodically. In general, AMiner performs a multi-strategy combining "from scratch", "on-the-fly" and "checking" together to solve the complex continuous name disambiguation problem. In this paper, we only introduce the principle of "on-the-fly" strategy under the assumption that the previously built profiles are correct, where the errors of the profiles are left to the "checking" strategy.

To tackle the problem, we first investigate how to find the top matched candidate for a given target paper. Straightforwardly, we can use the traditional feature-engineering methods to estimate the matching probability between each candidate and the target paper, and then return the top matched candidate. However, these methods are devoted to exactly matching the tokens between a paper and a person, which is too rigid and cannot handle the cases with similar semantics but different tokens. The widely used representation-based models [2], [46] can capture the soft/semantic matches through learning low-dimensional dense embeddings, but they may contrarily hurt the performance of exact matching due to the highly compressed embeddings. For example in Figure 1, if only depending on the semantics of learned embeddings, we can infer that both of the candidates are interested in social network mining. However, it is apparent that the exact matches of the coauthor names or words, e.g., "Jie Tang", "Juanzi Li", "social", "network" between the target paper and the right person are more than those of the wrong person. Thus, a challenge is posed: *how to capture both the exact matches and the soft matches in a principled way?* Simultaneously, the effects of different fields are different. For example, the two matched coauthors in the right person make it significantly more confident than the wrong person with only one matched coauthor, compared with the matches in other fields. Besides, each person publishes multiple papers, which also take different effects. For example in Figure 1, in the papers of the right person, the effect of the second similar paper may be diluted by the first irrelevant one if combining all papers. Thus, *an effective way to distinguish the effects of different fields of the attributes and different instances of the published papers is worth studying.*

After obtaining the top matched candidate, we need to decide whether to assign the top matched candidate or NIL candidate to the target paper. The NIL problem is widely studied in entity linking, a similar problem that aims at linking the mentions extracted from the unstructured text to the right entities in a knowledge graph. We can

adopt the similar idea to assign the NIL candidate to a target paper if the score of the top matched person is smaller than a NIL threshold [8], [32] or if the top matched person is predicted as NIL by an additional classifier [27]. Essentially, the first process of finding the top matched candidate tries to keep the relative distances between the right and the wrong persons of each target paper, and the later process of assigning the top matched candidate or not devotes to optimize the absolute positions among top matched candidates of all target papers. Intuitively, the two processes can influence each other, and the errors of each process can be corrected by their interactions. However, *none of the existing NIL solutions are aware of this and it is not clear how to correct the errors by the interactions between the two processes.*

To this end, in AMiner, we propose a joint model CONNA that consists of a matching component and a decision component to solve CONtinuous Name Ambiguity, i.e., name disambiguation on the fly, where "on the fly" emphasizes the solved problem in the paper is different from name disambiguation "from scratch". In the model, the matching component adopts an interaction-based deep learning model plus a kernel pooling strategy to capture both the exact and soft matches between a target paper and a candidate person and also a multi-field multi-instance strategy to distinguish the effects of different attributes and different instances of papers. The decision component is trained on the similarity embeddings learned by the matching component, to further decide whether a top matched person is the right person or not. In addition, the errors of the proposed model can be self-corrected through jointly fine-tune the two components by reinforcement learning. To summarize, the main contributions include:

- We propose CONNA consisting of a multi-field multi-instance interaction-based matching component and a decision component to address the problem of continuous name disambiguation. With jointly fine-tuning of the two components by reinforcement learning, the errors of the two components can be self-corrected.
- Experimental results on two large name disambiguation datasets show that CONNA compares favorably decision accuracy (+1.21%-19.84% in terms of F1) and matching accuracy (+ 3.80%-49.90% in terms of HR@1) against the baselines methods. CONNA is deployed on AMiner to assign papers on the fly now.

2 PROBLEM FORMULATION

We introduce the definitions and the problem in this section.

Definition 1: **Paper.** We denote a paper as p associated with multiple fields of attributes, i.e., $p = \{A_1, \cdots, A_F\}$, where $A_f \in p$ represents the f-th attribute such as authors' names and affiliations, title, keywords, venue and so on.

Definition 2: **Target paper-author pair.** Given a paper p with one of its authors denoted by a, we define a target paper-author pair as $\langle p, a \rangle$, where p is the target paper and a is the target author to be disambiguated. We abbreviate a target paper-author pair as a target pair henceforth.
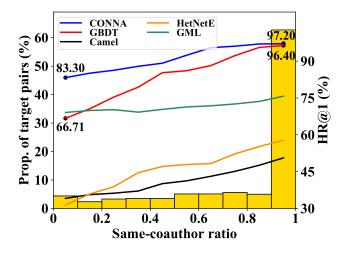
Fig. 2. Distribution of the same-coauthor ratio and the corresponding matching performance. Yellow bar: Distribution of the same-coauthor ratio of the target pairs. Lines: HR@1 performances of different methods.

Definition 3: **Candidate Persons.** Given a target pair $\langle p, a \rangle$, the corresponding candidate persons C are those who are closely related to the target pair $\langle p, a \rangle$. Each candidate person $c_l \in C$ is composed of multiple papers, i.e., $c_l = \{p_1, \cdots, p_{n_l}\}$, where each paper $p_t = \{A_1, \cdots, A_F\}$ and n_l is the number of papers published by c_l.

For a target pair $\langle p, a \rangle$, to find the right person from its candidate persons C, a straightforward way is to compare the coauthors' names of a in p with the coauthors' names of each candidate person in C^1. The assumption is the more overlaps between the coauthors' names, the more likely the candidate is the right author of p. The similar idea is adopted in [20], which found that if only using the users' names, 56% same users with different accounts across the social networks can be correctly linked together. However, how can the names take effect in identifying the right person for the target pairs?

To answer the question, we collect 100,000 target pairs from AMiner. For each target pair $\langle p, a \rangle$, we collect its candidate persons (Cf. Section 3.1 for candidate generation details) and calculate the same-coauthor ratio:

$$\text{Same-coauthor ratio} = \frac{\max_{c \in C} S_c - \text{second}_{c \in C} S_c}{\max_{c \in C} S_c - \min_{c \in C} S_c}, \quad (1)$$

where S_c is the number of the same coauthors of a in p with the candidate c. Same-coauthor ratio reflects the gap between the most similar candidate and the second similar candidate. The denominator is to normalize the gap calculated for different candidate lists into the same scale. It will be easier to distinguish the right person from the other candidates when the same-coauthor ratio is larger.

Then we plot the distribution of the same-coauthor ratio for all the target pairs in Figure 2, where X-axis indicates

1. The names are treated as strings to be compared with each other.

the same-coauthor ratio of a target pair, and Y-axis on the left denotes the proportion of the target pairs with a certain same-coauthor ratio. From the figure, we can see that although 62.72% target pairs have large same-coauthor ratios, there are still 14.59% target pairs having small same-coauthor ratios. The coauthor-related features will hardly take effect when dealing with the target pairs with small same-coauthor ratios. For these target pairs, it is also not easy to leverage other features except the coauthor features.

To verify the above hypothesis, we estimate the probability of matching each candidate person to the target pair by GBDT based on several features such as the literal similarities between the title, venue, or the affiliations of the target pair and those of a candidate person besides the coauthor-related features, then evaluate whether the top matched candidate is the right person or not and show the evaluated metric, top 1 Hit Ratio (i.e., HR@1 on the right Y-axis) for different ranges of the same-coauthor ratio in Figure 2. Clearly, we can see that the performance of GBDT decreases dramatically with the decrease of the same-coauthor ratio. The evaluated HR@1 is 66.71% when the same-coauthor ratio is within (0, 0.1), but is 96.40% within (0.9,1.0). The results indicate that when the coauthors of the target pair and the right person are not similar, it is also difficult for feature-engineering methods to capture the similarities of other attributes. Thus, a more promising way to match each candidate with the target pair is required.

In addition to find the top matched candidate, we also need to consider the situation when no right person exists, which is usually ignored by existing author identification tasks [2], [46]. Suppose an academic system establishes a profile for a researcher only if she/he has published at least one paper, a lot of papers written by the new researchers who publish papers for the first time, cannot be assigned to any existing person in the system. Thus, the right person should be either a real person or a non-existing person. In summary, the problem is defined as:

Problem 1: **Disambiguation on the fly.** Given a training set $\mathcal{D} = \{(\langle p, a \rangle, C)\}$, for each target paper-author pair $\langle p, a \rangle$ and the corresponding candidate persons C, the right person c^* can be either a real person in C denoted by c^+ or a non-existing person denoted by NIL, and other persons except c^* in C are the wrong persons denoted by $\{c^-\}$. The target is to learn a predictive function

$$\mathcal{F} : \{(\langle p, a \rangle, C)\} \to \{c^*\} \quad (2)$$

to assign a target paper-author pair to its right person.

In our problem, a is usually used to select candidate persons and p is used to extract features to match the candidates. To simplify the problem, we assume the historical papers assigned to the candidates are correct. However, historical errors cannot be avoided. Thus, we design an independent model to check and correct the historical assignments repeatedly. The study is left in the future.

3 CONNA

In this section, we first give an overview of the end-to-

end framework and then introduce the matching component which is to match the most possible candidate to the target pair and the decision component which is to decide whether to assign the top matched candidate to the target pair or not respectively. Finally, we introduce how to self-correct the errors of the two components by jointly fine-tuning them through reinforcement learning.

3.1 Overview

At first, given a target pair $\langle p, a \rangle$, the candidate persons C are the persons having the relevant names with the target author a. We define the relevant names as simple variants of a's name, including moving the last name to the first and keeping the initials of the names except for the last name. For example, the variants of "Jing Zhang" include "Zhang Jing", "J Zhang" and "Z Jing". For annotating a dataset for training and evaluating the models of name disambiguation, this simple candidate generation strategy can already result in enough challenging candidates.

The whole process of name disambiguation is divided into offline training and online predicting, which is shown in Figure 3. During the offline training process, we firstly train a matching component to estimate the probability of matching each candidate to the target pair and make the matching probability of the right person higher than those of the wrong persons for each target pair. The matching component constructs the training data from $\mathcal{D} = \{(\langle p, a \rangle, C)\}$ as a set of triplets $\mathcal{D}^r = \{(\langle p, a \rangle, c^+, c^-)\}$, where $\langle p, a \rangle$ is the target paper-author pair, c^+ is the real right person and c^- is a wrong person from the candidates. The objective is to make $\langle p, a \rangle$ closer to c^+ than to c^-. Then, we train a decision component to accept each sample $(\langle p, a \rangle, \hat{C}) \in \hat{\mathcal{D}}$ as the input and output a label $\hat{y}$ for the top matched person $\hat{c} \in \hat{C}$, where $\hat{C}$ is ranked by the trained matching component, $\hat{y} = 1$ indicates $\hat{c}$ is the right person and $\hat{y} = 0$ indicates $\hat{c}$ is the wrong person. We construct the training data D^c for the decision component by extracting $(\langle p, a \rangle, c^+)$ as the positive instance (i.e., $y = 1$) and $(\langle p, a \rangle, \hat{c}^-)$ as the negative instance (i.e., $y = 0$) from each sample $(\langle p, a \rangle, \hat{C})$, where $\hat{c}^-$ indicates the top matched wrong person in C. Finally, we fine-tune the matching component based on the feedback (i.e., error cases) of the decision component, and then fine-tune the decision component based on the updated output of the matching component. Essentially, the matching component tries to keep the relative distances between the right and the wrong persons of each target pair, and the decision component devotes to optimize the absolute positions between the top matched persons of all the target pairs found by the matching component.

During the online predicting process, to disambiguate a target pair $\langle p, a \rangle$, the matching component firstly finds out the top matched candidate person $\hat{c}$, then based on the similarity features $\phi(\langle p, a \rangle, \hat{c})$ output by the matching component, the decision component will predict the label $\hat{y}$ for $\hat{c}$ and finally assign the person c^* to $\langle p, a \rangle$, where $c^* = \hat{c}$ if $\hat{y} = 1$ and $c^* = \text{NIL}$ otherwise.
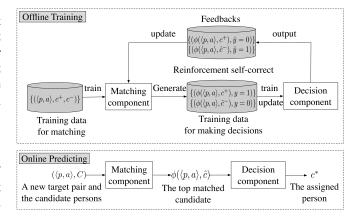

Fig. 3. The whole framework of training and predicting.

3.2 Matching

Basic Profile Model (BP). Let's imagine how humans assign a paper to a person. The humans usually browse all the papers published by the person to understand her/his affiliation, overall research interest, and frequently collaborated authors, then comparing them with those of the paper. In other words, humans directly compare the person's profile with the target pair, which can guide us to build our model. Thus, we name the model as the basic profile model. Specifically, we merge all the attributes of a paper and divide them into a set of tokens to represent the paper, and then merge the tokens of all the papers of a person into a unified set of tokens to represent the person's profile. Based on the token-based representations of the target paper and the person, we can estimate the similarity between them. Note a complete author name or a word in titles, keywords, venues and affiliations is viewed as a token.

Some metrics such as Jaccards Coefficient [30] and cosine similarity [30] can easily capture the exact matches. However, they suffer from the sparsity of the token-based representations. For example, the similarity is zero if two representations do not contain any same tokens, even if they are semantically similar. On the other hand, recently, some representation-based models [12], [14] can successfully capture the soft/semantic similarities, as they embed the high-dimensional sparse features into low-dimensional dense representations. Through training on the labeled data, the model can reduce the distance between the semantically similar inputs in the low-dimensional space. However, these models may suffer from the problem of semantic drift. For example, two token-based representations with many overlapped tokens may become dissimilar after being embedded by the model, as the global representation may dilute the effect of the exact same tokens by other different tokens. In summary, the above two types of methods are good at either exact matching or soft matching. To capture both the exact and soft matches, we adopt the interaction-based models [5], [12], [43] widely used in information retrieval. The interaction-based models first build a similarity matrix between each candidate person and the target pair and then apply an aggregation function to extract features
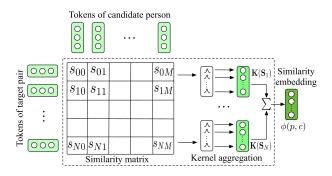
Fig. 4. The basic profile model.

Fig. 5. The multi-field profile model.

from the matrix. These models avoid learning the global representations, thus can reduce the issue of semantic drift.

Similarity Matrix. We represent the matches between each candidate and the target pair as a similarity matrix $\mathbf{S}$, with each element $\mathbf{S}_{ij}$ standing for the basic interaction, i.e., the cosine similarity $\mathbf{S}_{ij} = \frac{\mathbf{p}_i \cdot \mathbf{c}_j}{||\mathbf{p}_i|| \cdot ||\mathbf{c}_j||}$ between $\mathbf{p}_i$ and $\mathbf{c}_j$, where $\mathbf{p}_i$ represents the embedding of the i-th token in the target pair $\langle p, a \rangle$ and $\mathbf{c}_j$ represents the embedding of the j-th token in the candidate person c, which can be pre-trained by Word2Vec [23], [18] or BERT [6].

Aggregation Function. For sentence matching, CNN [12], [25] and RNN [39] are widely used as aggregation functions to extract matching patterns from the similarity matrix. However, different from sentence matching, title, keywords, venue and affiliation are all short text. We need to pay more attention to the occurrence of the exact same or semantically similar tokens. Thus we adopt an RBF kernel aggregation function [43] to extract features. Specifically, the i-th row $\mathbf{S}_i = \{S_{i0}, \cdots, S_{iM}\}$ of the similarity matrix — the similarities between the i-th token of the target pair and each token of the candidate person, is transformed into a feature vector $\mathbf{K}(\mathbf{S}_i)$, with each of the k-th element $K_k(\mathbf{S}_i)$ being converted by the k-th RBF kernel with mean μ_k and variance σ_k. Then the feature vectors of all the tokens in the target pair are summed up into the final similarity embedding $\phi(\langle p, a \rangle, c)$, i.e.,

$$\phi(\langle p, a \rangle, c) = \sum_{i=1}^{N} \log \mathbf{K}(\mathbf{S}_i), \qquad (3)$$

$$\mathbf{K}(\mathbf{S}_i) = \{K_1(\mathbf{S}_i), \cdots, K_K(\mathbf{S}_i)\}, \qquad (4)$$

$$K_k(\mathbf{S}_i) = \sum_{j=1}^{M} \exp\left[-\frac{(S_{ij} - \mu_k)^2}{2\sigma_k^2}\right]. \qquad (5)$$

The kernel with $\mu = 1$ and $\sigma \to 0$ only considers the exact matches between tokens, and others, e.g., with $\mu = 0.5$, counts the number of tokens in the candidate person whose similarities to a queried token in the target paper are close to 0.5. Thus, the kernel aggregation not only emphasizes the effect of exact matching but also captures the soft matches. Figure 4 illustrates the model.

Multi-field Profile Model (MFP). The basic profile model does not distinguish different fields of attributes but groups them together. However, it is not necessary to compare
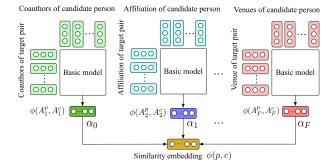
different attributes, such as comparing authors with venues. Moreover, it takes more effect to compare coauthor names than other attributes. So we build a basic profile model on each field of the attributes respectively, i.e. different attributes are not allowed to be compared, then aggregate the similarity embeddings together by the corresponding attention coefficients estimated by:

$$\alpha_f = \frac{\exp(w\phi(A_f^p, A_f^c) + b)}{\sum_f \exp(w\phi(A_f^p, A_f^c) + b)}, \qquad (6)$$

$$\phi(\langle p, a \rangle, c) = \sum_f \alpha_f \phi(A_f^p, A_f^c),$$

where $\phi(A_f^p, A_f^c)$ denotes the similarity embedding between A_f^p and A_f^c with A_f^p being the f-th field of p and A_f^c being that of the candidate person c. Notations w and b denote the parameters. The model is named as multi-field profile model and is shown in Figure 5.

Multi-field Multi-instance Model (MFMI). A person usually publishes multiple papers. Some persons even publish papers of multiple topics on multiple fields of venues and collaborate with multiple communities of persons. In this scenario, a target paper can be only similar to a small part of a person's diverse profile, but is totally irrelevant to other parts of the profile. However, the multi-field profile model may dilute the similarity with this small part when summing the similarities with all the tokens in a person's profile together by Eq.(5). To reduce the impact from the irrelevant papers, we build a multi-field model between the target pair and each published paper of the candidate person, and then aggregate the resultant similarity embeddings of all the published papers by their corresponding attention coefficients, which are estimated the same as Eq.(6). The model is named as the multi-field multi-instance model and is shown in Figure 6.

The Combination Model (CONNA^r). Essentially, the multi-field profile model captures the global similarities between the target pair and a person's profile, while the multi-field multi-instance model considers the local similarities between the target pair and each of the papers published by a person. Both of them can be explained intuitively, thus we can combine their output similarity embeddings together as the final feature embedding. We summarize different component variants in Table 1.
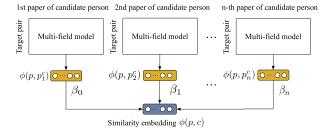
Fig. 6. The multi-field multi-instance model.

Loss Function. We use the triplet loss function to optimize the matching component. Similar ideas has been also used in [2], [46], [47]. Let $\mathcal{D}^r$ be a set of triplets with each triplet denoted as $(\langle p, a \rangle, c^+, c^-)$, where c^+ is the right person of the target pair $\langle p, a \rangle$ and c^- is a wrong person sampled from the candidates, the triplet loss function $\mathcal{L}(\Theta)$ is defined as:

$$
\begin{aligned}
\mathcal{L}(\Theta) &= \sum_{(\langle p,a \rangle, c^+, c^-) \in \mathcal{D}^r} \mathcal{L}_\Theta(\langle p, a \rangle, c^+, c^-) \\
&= \sum_{(\langle p,a \rangle, c^+, c^-) \in \mathcal{D}^r} \max\{0, g(\phi(\langle p,a \rangle, c^-)) - g(\phi(\langle p,a \rangle, c^+)) + m)\},
\end{aligned}
\tag{7}
$$

where g is defined to be a non-linear function to transform the similarity embedding ϕ into a one-dimension matching score that can be compared between the positive pair $(\langle p, a \rangle, c^+)$ and the negative pair $(\langle p, a \rangle, c^-)$. Notation Θ indicates the parameter of the matching component and $m > 0$ is a margin enforcing a distance between positive pairs and negative pairs. We optimize the triplet loss instead of directly optimizing the cross-entropy loss between the output matching score and the true label, as we aim at finding the top matched candidate from all the candidates for each target pair, thus the objective should be keeping a relative order within the candidate persons of each target pair instead of keeping a global order among all the (p, c) pairs. The triplet loss is more direct and close to our objective than the cross-entropy loss.

3.3 Decision

The decision component is built upon the output of the matching component to identify the right person, who can be either the top matched real person or NIL. The candidate persons C of each sample $(\langle p, a \rangle, C) \in \mathcal{D}$ are ranked into $\hat{C}$ based on the matching probabilities estimated by the matching component. Note for the samples with $c^* = c^+$, the real right person c^+ may be ranked the first or not. Then the decision component is trained to predict the first ranked person $\hat{c} \in \hat{C}$ to be a right person (i.e., $\hat{y} = 1$) or a wrong person (i.e., $\hat{y} = 0$). To achieve the goal, we construct the training data $\mathcal{D}^c$ from the ranked dataset $\hat{\mathcal{D}} = \{(\langle p, a \rangle, \hat{C})\}$. Specifically, from each sample $(\langle p, a \rangle, \hat{C})$, we extract $(\langle p, a \rangle, c^+)$ as the positive instance (i.e., $y = 1$) and extract $(\langle p, a \rangle, \hat{c}^-)$ as the negative instance (i.e., $y = 0$), where $\hat{c}^-$ indicates the first ranked wrong person in C. In another words, the positive instances are only extracted from the samples with $c^* = c^+$, while the

TABLE 1
Matching component variants of CONNA.

Component variants	Key idea
Basic Profile (BP)	The basic interaction-based model
Multi-field Profile (MFP)	Build BP for each field
Multi-field Multi-instance (MFMI)	Build MFP for each instance
CONNA^r	Combine MFP and MFMI

negative instances are extracted from both the samples with $c^* = c^+$ and the samples with $c^* = $ NIL. For an instance $(\langle p, a \rangle, c)$, we use the similarity embedding $\phi(\langle p, a \rangle, c)$ output by the matching component as its feature. Thus, $\mathcal{D}^c = \{(\phi(\langle p, a \rangle, c^+), y = 1)\} \cup \{(\phi(\langle p, a \rangle, \hat{c}^-), y = 0)\}$. Then we train a multi-layer perceptron $h(\Phi)$:

$$
h(\Phi) : \{\phi(\langle p, a \rangle, c)\} \rightarrow \{y\},
\tag{8}
$$

where y is the label of the instance $(\langle p, a \rangle, c)$, whose value equals 1 if $(\langle p, a \rangle, c)$ is a positive instance and 0 otherwise.

3.4 Reinforcement Self-correction

We finally fine-tune the two components by jointly training them to correct their errors by themselves. The matching component can be viewed as the generator to generate the ranking list. Without the decision component, the triplet loss in Eq.(7) is used to measure whether the ranking list is good or not. However, as the final objective is to determine whether the top ranked candidate is the right person or not, the triplet loss is not enough to verify the effect. Fortunately, we can use the prediction result of the top ranked candidate by the decision component as the delayed feedback to the ranking results of the matching component. Specifically, we can punish the ranking list with the wrongly predicted top candidate and reward the ranking list with the correctly predicted top candidate. Then based on the reward we update the matching component, expecting the ranking lists generated by the matching component to the decision component are more accurate. Followed by the motivation, we propose fine-tuning the two components via reinforcement learning. Specifically, the objective is to maximize the expected reward of the ranking lists generated by the matching component:

$$
J(\Theta) = \sum_{(\langle p,a \rangle, \hat{C})) \in \hat{\mathcal{D}}} p_\Theta(\langle p, a \rangle, \hat{C}) R(y, \hat{y}),
\tag{9}
$$

where $\hat{\mathcal{D}}$ is the ranked training data, $p_\Theta(\langle p, a \rangle, \hat{C})$ is the probability of generating the ranking list $\hat{C}$ of the target pair $\langle p, a \rangle$ by the matching component, and $R(y, \hat{y})$ is the reward function defined as follows:

$$
R(y, \hat{y}) = \begin{cases} 1 & \hat{y} = y; \\ 0 & \text{otherwise.} \end{cases}
\tag{10}
$$

where $\hat{y}$ is the predicted label for the top-ranked candidate $\hat{c}$ of $\hat{C}$ and y is the ground truth label. The defined reward function encourages the matching component to float the

---

Algorithm 1: Reinforcement Joint Training

Input: A training set $\mathcal{D} = \{(\langle p, a \rangle, C)\}$.
Output: A matching component and a decision component
parametrized by Θ and Φ respectively.
1 Build $\mathcal{D}^r = \{(\langle p, a \rangle, c^+, c^-)\}$ from $\mathcal{D}$;
2 Pre-train Θ of the matching component on $\mathcal{D}^r$;
3 Rank $\mathcal{D}$ by the matching component to generate $\hat{\mathcal{D}}$;
4 Build $\mathcal{D}^c = \{(\phi(\langle p, a \rangle, c)), y)\}$ from $\hat{\mathcal{D}}$;
5 Pre-train Φ of the decision component on $\mathcal{D}^c$;
6 **repeat**
7   **for** $(\langle p, a \rangle, \hat{C}) \in \hat{\mathcal{D}}$ **do**
8   Predict $\hat{y}$ for $\hat{c}$ by the decision component;
9   Calculate $R(y, \hat{y})$ by Eq.(10);
10   Calculate $\nabla_\Theta J(\Theta)$ by Eq.(11);
11       $\Theta \rightarrow \Theta + \mu \nabla_\Theta J(\Theta)$, where μ is the learning rate;
12   Re-rank $\mathcal{D}$ to generate $\hat{\mathcal{D}}$ by the matching component;
13   Re-generate $\mathcal{D}^c$ from $\hat{\mathcal{D}}$;
14   Update Φ of the decision component on $\mathcal{D}^c$;
15 **until** *Convergence*;

---

right person at the top and push the wrong person away from the top. The policy gradient algorithm [34] is adopted to optimize the expected reward in Eq.(9), whose gradient is calculated as:

$$\nabla_\Theta J(\Theta) = \sum_{(\langle p,a \rangle, \hat{C}) \in \hat{\mathcal{D}}} R(y, \hat{y}) \nabla p_\Theta(\langle p, a \rangle, \hat{C}),$$

$$\simeq \sum_{(\langle p,a \rangle, \hat{c}, c^-) \in \mathcal{D}^r} R(y, \hat{y}) \nabla \mathcal{L}_\Theta(\langle p, a \rangle, \hat{c}, c^-). \quad (11)$$

Since the probability of a ranking list $\hat{C}$ is not easy to be estimated, we transform $\hat{C}$ into a set of triplets, with each triplet including the target pair $\langle p, a \rangle$, the top ranked candidate $\hat{c} \in \hat{C}$ and a negative candidate $c^- \in \hat{C}$. Then the loss of a triplet in Eq.(7) is calculated and the losses of all the triplets are summed up to approximately measure the ranking performance of $\hat{C}$. Thus, the gradient $\nabla p_\Theta(\langle p, a \rangle, \hat{C})$ is approximated by $\nabla \mathcal{L}(\langle p, a \rangle, \hat{c}, c^-)$ of all the triplets in $\hat{C}$. Then the parameters Θ of the matching component can be updated by the gradient. After the matching component is tuned, the decision component is also updated based on the updated similarity embeddings output by the matching component. Algorithm 1 illustrates the joint training process, where we firstly pre-train the matching component and the decision component, and then jointly fine-tune the two components together.

4 EXPERIMENT

All codes and data used in the paper are publicly available[2].

4.1 Experimental Settings

4.1.1 Datasets

We evaluate CONNA on two name disambiguation datasets:

OAG-WhoIsWho[3]: Is the largest human-annotated name disambiguation dataset so far, which is consist of 608,363 papers belonging to 57,138 persons of 642 common names. Existing work either leverage the disambiguating results by algorithms in some well-known academic websites such as Scopus [28] CiteSeerX [45], Web of Science [1] and PubMed [38], or annotate a much smaller datasets by human beings, such as 8,453 [9], 6,921 [15], 7,528 [35] and 2,946 annotated persons [24]. Compared to the most popular KDD Cup 2013 challenge dataset, the OAG-WhoIsWho is also superior to it both in quantity (608,363 vs 424,384 in terms of the number of papers) and quality (fully human-labeled vs partially human-labeled). We annotate the dataset as follows. From the AMiner system, we choose 642 highly ambiguous names, create the relevant names by the candidate generation strategy in Section 3.1 and select all the authors for each name, collect all the papers assigned for each author and extract title, authors, organizations, keywords and abstract for each paper. We also collect all the unassigned papers for each name from AMiner. Since the assigned papers may be wrongly assigned and the papers are not fully assigned, additional efforts are needed to clean and reassign the papers. First, we clean the dataset by removing the wrongly assigned papers or splitting the papers of an author into different clusters. Second, we annotate the unassigned papers or merge the papers of two authors. We aim to clean the dataset as much as possible but increase the highly reliable assignments. According to the purpose, we only hire one annotator to perform the cleaning step, but hire three annotators to perform the assignment step respectively and then obtain the final results by majority voting their annotations. Besides, an annotation tool is developed to recommend highly reliable removing, splitting, assigning or merging operations to the annotators to simplify the human annotation process[4].

KDD Cup [29]: Is the dataset used in the KDD Cup 2013 challenge 1 to address name disambiguation problem. We collect the training data containing 3,739 authors and 123,447 papers, as only the training labels are published. We only use title, organizations, keywords and abstract as features, but ignore coauthor names. As shown in Figure 7(a), the distribution of same-coauthor ratio is extremely skewed. According to Eq.(1), same-coauthor ratio equalling 1 means the second similar candidate and the least similar candidate have the same number of same-coauthors with the target pair. In another word, the most similar candidate is significantly different from all the other candidates when only considering the coauthor name features. Thus, 98% target pairs holding 1.0 same-coauthor ratio means only using the coauthor names can correctly assign 98% target pairs. In fact, when considering the coauthor name feature, any baselines including our model can easily achieve approximate 99% HR@1. Thus, for increasing the difficulty, we ignore coauthor names on this dataset.
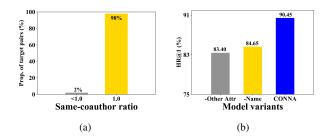

Fig. 7. (a) Distribution of the same-coauthor ratio on KDD Cup dataset; (b) The effects of different attributes.

4.1.2 Comparison Methods

Matching Component. To evaluate the matching performance, we compare feature engineering-based GBDT and three embedding-based models:

GBDT: Is a widely used model to solve KDD Cup 2013 challenge-1 [7], [19], [48]. We train a GBDT model to estimate a matching probability between each candidate and the target pair. The extracted features for GBDT are shown in Table 2. As the model can directly predict a label for each candidate, it also be used for deciding to assign the most matched candidate to the target pair if its label is 1.

Camel [46]: Is a representation-based model. Given a triplet $(\langle p, a \rangle, c^+, c^-)$, it first represents $\langle p, a \rangle$ by p's title, and represents c^+ and c^- by their identities. Then it calculates the matching scores for both $(\langle p, a \rangle, c^+)$ and $(\langle p, a \rangle, c^-)$, and finally optimizes the difference between their matching scores.

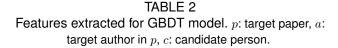*HetNetE* [2]: Is similar as Camel except that $\langle p, a \rangle$ is represented by all its attributes.

GML [47]: Is a representation-based model to identify whether two papers are written by the same person through optimizing a triplet loss. The model accepts the pre-trained embeddings of all the tokens in a paper as input and output an embedding for the paper. We represent a person by averaging all his/her papers' embeddings.

Decision Component. To evaluate the performance of the decision component, we compare two strategies:

Threshold [8]: Picks the top matched person whose score is lower than a threshold as NIL, where the threshold is determined as the value when the best accuracy is obtained on a validation set. We use the same matching model as our proposed method to obtain the top matched persons.

Heuristic Loss [4]: Unifies the NIL decision and the matching process by incorporating the costs of assigning a paper to a wrong NIL person or assigning an unlinkable paper to a wrong existing person into the loss function of ranking the wrong person before the right person. NIL is inserted as an additional candidate person for each paper. The representations of p and c which are made in the same way as GML, are concatenated as the input of a neural network to produce their matching score. When $c = $ NIL, the representation of c is not included.

Variants of Our Model. We also compare different variants where CONNA^r(BP), CONNA^r(MFP),

TABLE 2
Features extracted for GBDT model. p: target paper, a: target author in p, c: candidate person.

No.	Feature description
1	The number of the papers of c
2	The number of the coauthors of a in p
3	The number of the coauthors of c
4	The number of the same coauthors between a and c
5	Ratio of the same coauthors between a and c in p's coauthor names
6	Ratio of the same coauthors between a and c in c's coauthor names
7	Frequency of a's affiliation in c's affiliations
8	Ratio of a's affiliation in c's affiliations
9	Cosine similarity between a's affiliation and c's affiliations
10	Jaccards similarity between a's affiliation and c's affiliations
11	Distinct number of venues of c
12	Frequency of p's venue in c
13	Ratio of p's venue in c
14	Cosine similarity between p's venue and c's venues
15	Jaccards similarity between p's venue and c's venues
16	Cosine similarity between p's title and c's titles
17	Jaccards similarity between p's title and c's titles
18	Distinct number of keywords in c
19	Frequency of p's keywords of c
20	Ratio of p's keywords in c
21	Cosine similarity between p's keywords and c's keywords
22	Jaccards similarity between p's keywords and c's keywords

CONNA^r(MFMI) and CONNA^r correspond to the variants in Table 1. CrossEntropy modifies CONNA^r by replacing the triplet loss with the cross-entropy loss, which can be directly used for deciding the assignments. CONNA trains CONNA^r plus a decision component once. CONNA+Finetune jointly trains the two components in CONNA.

4.1.3 Evaluation Settings

For each dataset, we randomly sample 20% persons for testing and divide the rest into training, which results in 45,711 authors for training and 11,427 authors for testing on OAG-WhoIsWho dataset, and 2,991 authors for training and 748 authors for testing on KDD Cup dataset. For each author in both training and testing data, we first sort their papers by the published year in ascending order. Then we choose the latest 20% papers as the author's unassigned paper and leave 80% papers as the author profile.

We first evaluate the matching of the candidate persons to the target pair, and further evaluate the decision of the top matched person as the right person or NIL.

Matching Evaluation. For evaluating the matching performance, we sample 10,000 target pairs from the training data. Each target pair paired with its right person composes a positive instance. We also sample 9 wrong persons paired with each target paper to compose 9 negative instances. The process results in 90,000 triplets for training. For testing, we sample 2,000 target pairs from the test data, where each one is associated with the right person and 19 wrong persons.

The wrong persons are sampled from the candidates. We follow the name variant strategy in section 3.1 to generate candidates on OAG-WhoIsWho. While for KDD Cup, names are so different that no candidates can be found by simply varying names. Instead, we calculate the Jaro-

Winkler similarity between a candidate's name and the target author, and select the candidates whose scores are larger than 0.5 as the wrong persons.

We use Hit Ratio at top k (HR@k) and mean reciprocal rank (MRR) as the metrics for evaluating whether the right person will be ranked at the top among all the candidates. Since there is only one right person for each target pair, HR@k measures the percentage of the candidate lists with the right person ranked before top k. MRR measures the average of reciprocal ranks of the right persons. Higher HR@k and MRR indicate better performance.

Decision Evaluation. We construct the training data for the decision component upon the output of the matching component. Specifically, we also use the 10,000 positive instances for the matching component as those for the decision component. Then we extract the target pairs and the corresponding top matched wrong persons to compose the negative instances. For testing, in addition to the 2,000 target pairs and the corresponding candidates including the right persons (i.e., positive sample $(\langle p, a \rangle, C)$ with $c^* = c^+$), we extract extra 2,000 target pairs and the corresponding candidates excluding the right persons (i.e., negative sample $(\langle p, a \rangle, C)$ with $c^* = \text{NIL}$). Conveniently, we remove the right person c^+ from each positive sample and create a negative sample by the remaining wrong persons. We count the number of true positive (tp), false negative (fn), true negative (tn) and false positive (fp) samples and then calculate precision, recall and f1:

$$
\text{tp} = |\{c^* = c^+ \text{ and } \hat{c} = c^+ \text{ and } \hat{y} = 1\}|, \quad (12)
$$
$$
\text{fn} = |\{c^* = c^+ \text{ and } \hat{y} = 0\}|,
$$
$$
\text{tn} = |\{c^* = \text{NIL} \text{ and } \hat{y} = 0\}|,
$$
$$
\text{fp} = |\{c^* = \text{NIL} \text{ and } \hat{y} = 1\} \cup
$$
$$
\{c^* = c^+ \text{ and } \hat{c} \neq c^+ \text{ and } \hat{y} = 1\}|
$$

where tp is the number of the positive samples, with the right persons ranked at the first (i.e., $\hat{c} = c^+$) and also predicted as the right persons (i.e., $\hat{y} = 1$). On the contrary, fn counts the positive samples with $\hat{y} = 0$. Notation tn denotes the number of negative samples with the first ranked persons predicted as the wrong persons (i.e., $\hat{y} = 0$), while fp counts the negative samples with $\hat{y} = 1$ and also counts the positive samples with the wrong persons ranked at the first (i.e., $\hat{c} \neq c^+$) but still predicted as the right persons (i.e., $\hat{y} = 1$). Since we aim at assigning the target pair to an existing right person and also assigning it to NIL if there is no right person, we calculate precision and recall for both the cases with $c^* = c^+$ and $c^* = \text{NIL}$:

$$
c^* = c^+ \; : \; \text{Pre.} = \frac{\text{tp}}{\text{tp} + \text{fp}}, \quad \text{Rec.} = \frac{\text{tp}}{\text{tp} + \text{fn}}; \quad (13)
$$
$$
c^* = \text{NIL} \; : \; \text{Pre.} = \frac{\text{tn}}{\text{tn} + \text{fn}}, \quad \text{Rec.} = \frac{\text{tn}}{\text{tn} + \text{fp}}.
$$

TABLE 3
Performance of the matching results (%).

Model	OAG-WhoIsWho			KDD Cup		
	HR@1	HR@3	MRR	HR@1	HR@3	MRR
Camel	41.20	62.00	55.00	44.62	67.19	59.44
HetNetE	46.00	67.00	60.24	51.06	77.44	66.41
GML	70.87	94.53	82.59	72.13	95.34	82.90
GBDT	87.30	98.10	92.71	84.18	92.09	89.59
CONNA^r(BP)	86.20	96.40	92.20	91.12	95.72	93.73
CONNA^r(MFP)	88.00	98.75	93.25	-	-	-
CONNA^r(MFMI)	89.45	98.40	93.82	91.45	95.80	94.03
CONNA	90.45	98.30	94.46	92.10	96.35	94.66
CONNA+Fine-tune	**91.10**	**98.45**	**94.86**	**92.60**	**96.71**	**94.95**

4.1.4 Implementation Details

We divide the attributes of a paper into two fields: coauthor names and other attributes including title, abstract, organizations and keywords, as coauthor names have no literal or semantic overlaps with other attributes. We pre-train an embedding for each author name and each word. Specifically, we use Word2Vec to train an embedding for an author name in the context of all the coauthors' names in a paper, and train an embedding for a word in the context of all the other occurred words in title, keywords, venue and affiliation. We set the dimension of the embedding as 100. To enable matrix operation, for each paper or candidate person, we restrict the maximal number of author names to 100, the maximal number of words to 500, and the maximal number of papers published by each person to 100.

The hyper-parameters of the RBF kernel functions are set the same as [43]. We use 11 RBF kernels, with the hyper-parameters $\mu = \{1, 0.9, 0.7, 0.5, 0.3, 0.1, -0.1, -0.3, -0.5, -0.7, -0.9\}$ and $\sigma = \{10^{-3}, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1\}$.

Function g in Eq.(7) is instantiated as a 3-layer MLP followed by a ReLU function which transforms a similarity embedding $\phi(\langle p, a \rangle, c)$ into a 1-dimensional score. Function h in Eq.(8) is also a 3-layer MLP which transforms a $\phi(\langle p, a \rangle, c)$ into 2-dimensional classification probabilities.

4.2 Performance Analysis

4.2.1 Matching Performance

Overall Matching Performance. Table 3 shows the matching performance of the proposed model, the model variants and the comparison methods on the two datasets OAG-WhoIsWho and KDD Cup. In terms of HR@1, the proposed CONNA+Fine-tune achieves 3.80% to 49.90% improvement over all the baseline methods.

Camel, HetNetE and GML are all representation-based deep learning models, which can capture the soft/semantic matches, but they will dilute the effect of the exact matches of tokens due to the global representations of the papers and persons. Among the three models, HetNetE uses all the attributes of a paper rather than the single title to represent a paper, which achieves better performance than
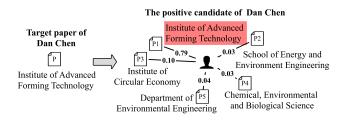
Fig. 8. Case study of multi-field effect.

Fig. 9. Case study of multi-instance effect.

Camel. Camel and HetNetE represent the candidate persons only based on their identities. Thus they suffer from the sparsity issue, i.e, the embeddings of the persons cannot be trained accurately if they publish few papers. GML avoids the sparsity issue through representing persons by their published papers. However, it is difficult to directly compare the embeddings of a long text (i.e, all the papers of a candidate person) and a short text (i.e., a target paper).

In the name disambiguation problem, the exact matches between tokens especially the matches between coauthor names are more important than the soft matches, thus although GBDT only captures the exact matches, it performs better than the representation-based models. The proposed interaction-based matching component in CONNA captures both the exact and the soft matches through comparing local representations of each token pairs instead of comparing the global representations of papers and persons. Specifically, the kernel aggregation function used in the matching component summarizes a frequency distribution of the exact matches and different kinds of soft matches, which can't dilute the effect of extract matches by the other soft matches. Thus, the proposed matching component performs better than all the comparison methods.

Compared with CONNA, the performance of CONNA+Fine-tune is further improved, as the decision component gives additional feedbacks to supervise the ranking of the matching component. The result indicates that through jointly fine-tuning of the two components, the errors of the matching component can be reduced.

Comparing the results on the two datasets, we can see that the advantage of our model over the feature engineering-based GBDT is much more significant on KDD Cup (+8.42% in HR@1) than OAG-WhoIsWho (+3.80% in HR@1). Since coauthor features are not used on the KDD Cup, the results indicate that CONNA can better capture

the semantics of the attributes except coauthor names.

Multi-field Effect.   We conduct an ablation study to analyze the effects of different modeling strategies on the matching component. Since only one field is used on the KDD Cup dataset, we analyze the effect of multi-fields on the OAG-WhoisWho dataset. From Table 3, we can see that $CONNA^r$(MFP) performs better than $CONNA^r$(BP) (improving 1.8% in terms of HR@1), which indicates that it is necessary to build the interaction-based models for different attributes separately and distinguish their effects.

We also investigate the effects of different fields by removing coauthor names and other attributes respectively based on the model CONNA. The experimental results in Figure 7(b) show that removing either coauthor names or other attributes performs significantly worse (-5.80%-7.05%, HR@1) than CONNA, which indicates that both coauthor names and other attributes impact the performance obviously. What's more, removing names is comparable to removing other attributes, which indicates that coauthor names are more important than all the other attributes on the task of name disambiguation.

Multi-instance Effect.   Table 3 also shows that on OAG-WhoisWho, $CONNA^r$(MFMI) performs better than $CONNA^r$(MFP) (+1.45% in terms of HR@1), which demonstrates the strength of distinguishing different papers of a person. HR@1 of $CONNA^r$(MFMI) is further improved by 1.00% if we combine the profile model $CONNA^r$(MFP) and the multi-instance model $CONNA^r$(MFMI) as CONNA. The result indicates that both the global similarity between the target paper and a candidate's whole profile, and the local similarities between the target paper and each paper of a candidate take effects on matching performance. The results on KDD Cup also present the advantages of multi-instances.

Interpretability of the Matching Component. We present some cases in Figure 8 and Figure 9 to demonstrate the interpretability of the proposed matching component. From Figure 8, we can see that although the number of the matched tokens between the target paper and the positive candidate person is less than that of the negative candidate person, the matched coauthors are more important than the matched words in titles and venues, because the attention α learned by our model for the matched coauthors on the positive candidate is 0.69, comparing with 0.31 learned for the matched titles and venues. And the attention learned on the negative candidate also emphasizes the matched coauthors. CONNA distinguishes different fields' effects by the attention, thus it can correctly identify the positive candidate, while the basic profile model $CONNA^r$(BP) wrongly returns the negative candidate as the most matched candidate, as it treats the matches in all the fields equally.

In Figure 9, we present the affiliation of "Dan Chen" in both the target paper and the positive candidate. It is shown that a paper of the positive candidate has the same affiliation with the target paper, and the corresponding attention β learned by our model for the paper is 0.79, while the values of β learned for other papers are much smaller than this

TABLE 4
Performance of the decision results (%).

Model	OAG-WhoIsWho						KDD Cup					
	Samples with $c^* = c^+$			Samples with $c^* = $ NIL			Samples with $c^* = c^+$			Samples with $c^* = $ NIL		
	Pre.	Rec.	F1	Pre.	Rec.	F1	Pre.	Rec.	F1	Pre.	Rec.	F1
GBDT	82.87	72.40	77.28	75.39	85.04	79.98	83.64	71.64	77.17	75.20	85.98	80.23
Threshold	79.33	57.60	66.38	66.47	**84.07**	74.24	74.89	71.00	72.90	72.43	76.20	74.27
Heuristic Loss	71.79	78.40	74.95	76.21	69.20	72.54	85.14	69.60	76.59	74.29	87.85	80.50
CrossEntropy	79.42	82.33	80.85	81.66	78.67	80.14	89.60	82.79	86.06	86.15	88.05	87.09
CONNA	79.53	89.87	84.38	88.35	76.87	82.21	88.44	**86.20**	87.31	**86.54**	88.73	87.62
CONNA+Fine-tune	**82.47**	**90.33**	**86.22**	**89.31**	80.80	**84.84**	**89.87**	85.73	**87.75**	86.36	**90.33**	**88.30**

TABLE 5
Average time cost(ms) of assigning each target pair.

Model	Feature Preparing	Matching	Decision
GBDT	183.34	–	3.61
CONNA	260.45	76.12	6.34

paper. CONNA distinguishes different papers' effects, thus it can correctly identify the positive candidate, while the basic profile model CONNA^r(BP) treats the matches in all the papers equally, which dilutes the effects of similar papers by the other irrelevant papers. The learned attentions for different fields and different papers both demonstrates the interpretability of the proposed matching component.

Matching Performance on Different Scenarios. We conduct additional experiments on the matching performance of different baselines and CONNA with different same-coauthor ratios on OAG-WhoIsWho dataset and present the results in Figure 2. We can see that HR@1 of the embedding-based models, i.e. Camel, HetNetE, GML and CONNA drop more slightly (drops from 6.63% to 18.73%) than feature-engineering based GBDT (drops more than 29.69%) when the same-coauthor ratio decreases from 1.0 to 0.1. This indicates that the embedding-based model can better capture the semantic matches when the coauthor features are week. Especially when the same-coauthor ratio is less than 0.1, the performance gap between CONNA and GBDT is significantly more than 16%. The result indicates that CONNA is more suitable to tackle the hard cases, i.e. the cases that are hardly predicted by similar coauthors.

4.2.2 Decision Performance

Table 4 shows the final decision performance of the proposed model and the comparison methods. Comparing with other methods, in terms of F1, the proposed joint model CONNA+Fine-tune achieves 1.69%-19.84% improvement on the samples with $c^* = c^+$ and 1.21%-14.03% improvement on the samples with $c^* = $ NIL. We evaluate the results on both of the samples as we aim at not only assigning the target papers to the right persons if they exist, but also assigning them to NIL if the right persons do not exist. The problem in this paper is not merely a matching or a classification decision problem, but can be solved by firstly

matching each candidate to the target paper p and then deciding whether the top matched person is right or not. Thus, we need to not only keep the relevant order within each candidate list, but also globally distinguish all the positive pairs from all the negative pairs.

GBDT and CrossEntropy only aim to optimize the global positions of all the $(\langle p, a \rangle, c)$ pairs, but ignore the relative order within each candidate list. Although the globally predicted probabilities can be used to compare the candidates of each target paper, the relative order is not directly optimized, leading to a lot of mistakes in the final results. Threshold can be viewed as a global optimization model, but merely uses a heuristic threshold to distinguish different complicated cases. Heuristic Loss incorporates the costs related to NIL into the original loss of ranking the wrong persons before the right persons, but it suffers from the heuristically configured weights of different costs.

CONNA first estimates the matching probability of each candidate to the target pair and then decides the top matched candidate. This two-step strategy which is widely adopted in entity linking [22], [27] is proved to be effective. Compared with CONNA, the performance of CONNA+Fine-tune is further improved, as some of the wrongly-predicted instances are gradually represented better to generate accurate similarity embeddings by the iteratively refined matching component, which will finally increase the number of rightly predicted instances. The result demonstrates that the errors of the decision component can be reduced through jointly fine-tuning of the two components.

Convergence Analysis. We plot the train/test loss of the matching component and the decision component with the increase of the joint training epochs. The results in Figure 10 show that the performance of the two components both decrease sharply at the beginning of the joint training and then gradually change stable, which indicate the convergence of CONNA+Fine-tune.

4.3 Online Deployment on AMiner

Table 5 presents the average time cost of assigning each target paper by the proposed CONNA model and the best baseline GBDT. We implement the experiments by Tensorflow and run the code on an Enterprise Linux Server with 40 Intel(R) Xeon(R) CPU cores (E5-2640 v4 @
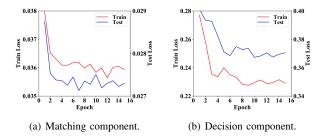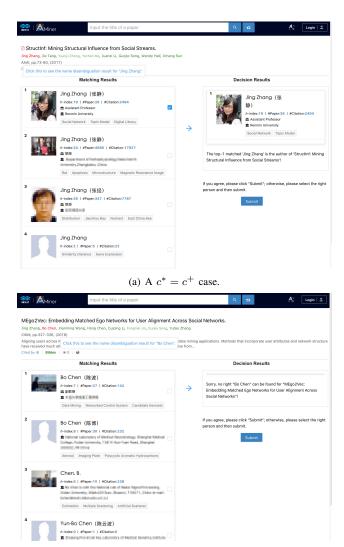
(a) Matching component. (b) Decision component.

Fig. 10. Convergence Analysis.

(a) A $c^* = c^+$ case.

(b) A $c^* = $ NIL case.

Fig. 11. A demo of disambiguation on the fly in AMiner.

2.40GHz and 252G memory) and 1 NVIDIA Tesla V100 GPU core (32G memory). Since GBDT is a classification model without the matching component, we only present the cost of the decision process, which uses the label of the top predicted candidate as the predictive result. From Table 5, we can see that CONNA is about $1.83\times$ slower than GBDT, which is mainly determined by the feature preparing process. Although CONNA performs much better than GBDT on both the ranking and the decision performance, from Figure 2, we can see for about 62.17% easy samples, i.e., the target pairs with the same-coauthor ratio larger than 0.9, the ranking performance of GBDT is comparable to CONNA, where the ranking performance directly determines the final decision performance of the top-1 candidates. Thus, to improve the online assignment efficiency meanwhile keeping the assignment performance, for each target pair, if its same-coauthor ratio is larger than 0.9, we directly apply GBDT to perform paper assignment, otherwise we apply CONNA to complete the task.

In addition, the online candidate selection is a little different from the offline name variant strategy explained in Section 3.1. To improve the recall of the online predicting as much as possible, we adopt ElasticSearch[5] to perform fuzzy search for similar candidates with each target author. Compared with this online fuzzy strategy, the offline candidate selection is more strict, as for annotating high-quality name disambiguation dataset, the simple name variant strategy can already produce enough challenging candidates. However, the fuzzy strategy may result in too many noisy candidates, which increase annotation efforts.

We develop a demo of disambiguation on the fly in AMiner[6], and show two screenshots of the demo in Figure 11. In the demo, users are allowed to search a paper by its title, then select the expected paper and click one author name to see the disambiguation results of the paper with the current name. Under the selected paper, we present the most matched candidates by the trained matching component in CONNA on the left, and show the decision result of the assigned person by the trained decision component in CONNA on the right. Figure 11(a) shows a case with $c^* = c^+$. We can see that our model can correctly match "Jing Zhang" from Renmin University for the author "Jing Zhang" in the paper "StructInf: Mining Structural Influence from Social Streams" at the top and then decide the top

5. https://www.elastic.co
6. http://na-demo.aminer.cn/

matched one as the final assigned person. Figure 11(b) shows a case with $c^* = $ NIL. Since "Bo Chen" of the paper "MEgo2Vec: Embedding Matched Ego Networks for User Alignment Across Social Networks" is a postgraduate student whose profile has not been established by AMiner, none of the existing "Bo Chen" should be assigned to the paper. Our model correctly assigns NIL to this case. Besides, since errors are still inevitable, we allow the users to provide feedback to our decision results. Specifically, users are allowed to directly "submit" the result if they agree with it, otherwise, they can choose another right person from the top matched persons. The feedback can be simply regarded as new training instances to update the decision performance at each step of the joint training.

5 RELATED WORK

This paper is related to the problems of name disambiguation from scratch, author identification and entity linking.

Name Disambiguation from Scratch. Much effort has been made to disambiguate names from scratch defined as:

given a set of papers written by the authors with similar name, it targets at partitioning all the papers into several disjoint clusters, with each of them corresponds to a real person. Existing work firstly represent papers by traditional feature engineering methods [3], [13], [35], [40], [42] or embedding models [26], [41], [45], [47] and then adopt a clustering algorithm such as hierarchical agglomerative clustering [3], [26], [41], [45], [47], K-means [40], DB-SCAN [13] or semi-supervised clustering [21] to partition these papers. Embedding models further include graph auto-encoder [47], heterogeneous GCN [26] and adversarial representation learning [41]. Continuous name disambiguation is formalized differently from the above problem, thus it can not be solved by the above methods.

Author Identification. Several works devote to anonymous author identification for a paper, which assume the authors of the target paper are unknown in a double-blind setting. For example, Chen et al. [2] and Zhang et al. [46] both optimize the difference between the right and the wrong authors. However, their models cannot be applied to unseen authors in the training set, as they only consider the identities of the authors. While we model authors' profiles, which do not depend on authors' identities. KDD Cup 2013 held an author identification challenge to solve the similar problem. However, the situation that no right person exists was not considered and all the participations devoted to feature-engineering methods [7], [48].

Entity Linking. Entity linking aims at linking the mentions extracted from the unstructured text to the right entities in a knowledge graph [31]. Feature-based [17] or neural models such as skip-gram [44], autoencoder [11], CNN [33], LSTM [16] are proposed to calculate the similarity between the context of a mention and a candidate entity. The NIL problem is widely studied in entity linking. The main solutions usually include the NIL threshold methods [8], [32] predicting the mention as unlinkable if the score of the top ranked entity is smaller than a NIL threshold, the classification methods [22], [27] which predict the unlinkable mentions by a binary classifier, and the unified models incorporating unlinkable mention prediction process into entity matching process [4], [10]. Different from above, we jointly train the NIL decision model and the candidate matching model to boost both of their performance.

6 CONCLUSION

This paper presents the first attempt to formalize and solve the problem of name disambiguation on the fly by considering different cases of assignments, in particular when a paper cannot be assigned to any existing persons in the system. We propose a novel joint model that consists of a matching component and a decision component, where a multi-field multi-instance interaction-based model is trained to match the candidates to each target paper, and then a classification decision model is trained to decide whether to assign the top matched candidate to the target paper or not. Through reinforcement joint fine-tuning, the two components can bootstrap each other and self-correct some

of their errors. The experimental results on the recent largest dataset for name disambiguation demonstrate that the proposed model performs significantly better than state-of-the-art baseline methods. The model has already been deployed on AMiner to disambiguate the online papers.

REFERENCES

[1] T. Backes. The impact of name-matching and blocking on author disambiguation. In *CIKM'18*, pages 803–812, 2018.

[2] T. Chen and Y. Sun. Task-guided and path-augmented heterogeneous network embedding for author identification. In *WSDM'17*, pages 295–304, 2017.

[3] Y. Chen and J. Martin. Towards robust unsupervised personal name disambiguation. In *EMNLP-CoNLL'07*, pages 190–198.

[4] K. Clark and C. D. Manning. Deep reinforcement learning for mention-ranking coreference models. In *ACL'16*, 2016.

[5] Z. Dai, C. Xiong, J. Callan, and Z. Liu. Convolutional neural networks for soft-matching n-grams in ad-hoc search. In *WSDM'18*, pages 126–134, 2018.

[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT'19*, 2019.

[7] D. Efimov, L. Silva, and B. Solecki. Kdd cup 2013-author-paper identification challenge: Second place team. In *KDD Cup 2013 Workshop*, page 3, 2013.

[8] S. Gottipati and J. Jiang. Linking entities to a knowledge base with query expansion. In *EMNLP'11*, pages 804–813, 2011.

[9] H. Han, L. Giles, H. Zha, C. Li, and K. Tsioutsiouliklis. Two supervised learning approaches for name disambiguation in author citations. In *JCDL'04*, pages 296–305, 2004.

[10] X. Han and L. Sun. A generative entity-mention model for linking entities with knowledge base. In *ACL'11*, pages 945–954, 2011.

[11] Z. He, S. Liu, M. Li, M. Zhou, L. Zhang, and H. Wang. Learning entity representation for entity disambiguation. In *ACL'13*, pages 30–34, 2013.

[12] B. Hu, Z. Lu, H. Li, and Q. Chen. Convolutional neural network architectures for matching natural language sentences. In *NIPS'14*, pages 2042–2050, 2014.

[13] J. Huang, S. Ertekin, and C. L. Giles. Efficient name disambiguation for large-scale databases. In *PKDD'06*, pages 536–544, 2006.

[14] P.-S. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. Heck. Learning deep structured semantic models for web search using clickthrough data. In *CIKM'13*, pages 2333–2338, 2013.

[15] I.-S. Kang, P. Kim, S. Lee, H. Jung, and B.-J. You. Construction of a large-scale test set for author disambiguation. *Information Processing & Management*, 47(3):452–465, 2011.

[16] N. Kolitsas and O.-E. Ganea. End-to-end neural entity linking. In *ACL'18*, 2018.

[17] J. Lehmann, S. Monahan, L. Nezda, A. Jung, and Y. Shi. Lcc approaches to knowledge base population at tac 2010. In *TAC*, 2010.

[18] B. Li, A. Drozd, Y. Guo, T. Liu, S. Matsuoka, and X. Du. Scaling word2vec on big corpus. *Data Science and Engineering*, 4(2):157–175, 2019.

[19] J. Li, X. Liang, W. Ding, W. Yang, and R. Pan. Feature engineering and tree modeling for author-paper identification challenge. In *KDD Cup 2013 Workshop*, page 5, 2013.

[20] J. Liu, F. Zhang, X. Song, Y.-I. Song, C.-Y. Lin, and H.-W. Hon. What's in a name?: an unsupervised approach to link users across communities. In *WSDM'13*, pages 495–504. ACM, 2013.

[21] G. Louppe, H. T. Al-Natsheh, M. Suszhanik, and E. J. Maguire. Ethnicity sensitive author disambiguation using semi-supervised learning. In *KESW'16*, pages 272–287, 2016.

[22] P. McNamee. Hltcoe efforts in entity linking at tac kbp 2010. In *TAC*, 2010.

[23] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *ICLR'13*, 2013.

[24] M.-C. Müller, F. Reitz, and N. Roy. Data sets for author name disambiguation: an empirical analysis and a new resource. *Scientometrics*, 111(3):1467–1500, 2017.

[25] L. Pang, Y. Lan, J. Guo, J. Xu, S. Wan, and X. Cheng. Text matching as image recognition. In *AAAI'16*, pages 2793–2799, 2016.

[26] Z. Qiao, Y. Du, Y. Fu, P. Wang, and Z. Y. c. Unsupervised author disambiguation using heterogeneous graph convolutional network embedding. In *IEEE BigData'19*, pages 910–919, 2019.

[27] L. Ratinov, D. Roth, D. Downey, and M. Anderson. Local and global algorithms for disambiguation to wikipedia. In *ACL'11*, pages 1375–1384, 2011.

[28] L. Reijnhoudt, R. Costas, E. Noyons, K. Börner, and A. Scharnhorst. 'seed+ expand': a general methodology for detecting publication oeuvres of individual researchers. *Scientometrics*, 101(2):1403–1417, 2014.

[29] S. B. Roy, M. De Cock, V. Mandava, S. Savanna, B. Dalessandro, C. Perlich, W. Cukierski, and B. Hamner. The microsoft academic search dataset and kdd cup 2013. In *KDD cup 2013 workshop*, pages 1–6, 2013.

[30] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., 1983.

[31] W. Shen, J. Wang, and J. Han. Entity linking with a knowledge base: Issues, techniques, and solutions. *TKDE'14*, 27(2):443–460, 2014.

[32] W. Shen, J. Wang, P. Luo, and M. Wang. Linking named entities in tweets with knowledge base via user interest modeling. In *KDD'13*, pages 68–76, 2013.

[33] Y. Sun, L. Lin, D. Tang, N. Yang, Z. Ji, and X. Wang. Modeling mention, context and entity with neural networks for entity disambiguation. In *IJCAI'15*, 2015.

[34] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *NIPS'00*, pages 1057–1063, 2000.

[35] J. Tang, A. C. Fong, B. Wang, and J. Zhang. A unified probabilistic framework for name disambiguation in digital library. *TKDE'12*, 24(6):975–987, 2012.

[36] J. Tang, L. Yao, D. Zhang, and J. Zhang. A combination approach to web user profiling. *TKDD'10*, 5(1):2, 2010.

[37] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su. Arnetminer: extraction and mining of academic social networks. In *SIGKDD'08*, pages 990–998, 2008.

[38] V. I. Torvik and N. R. Smalheiser. Author name disambiguation in medline. *TKDD'09*, 3(3):11, 2009.

[39] S. Wan, Y. Lan, J. Xu, J. Guo, L. Pang, and X. Cheng. Match-srnn: Modeling the recursive matching structure with spatial rnn. In *IJCAI'16*, 2016.

[40] F. Wang, J. Tang, J. Li, and K. Wang. A constraint-based topic modeling approach for name disambiguation. *Frontiers of Computer Science in China*, 4(1):100–111, 2010.

[41] H. Wang, R. Wan, C. Wen, S. Li, Y. Jia, W. Zhang, and X. Wang. Author name disambiguation on heterogeneous information network with adversarial representation learning. In *AAAI'20*, pages 238–245, 2020.

[42] X. Wang, J. Tang, H. Cheng, and S. Y. Philip. Adana: Active name disambiguation. In *ICDM'11*, pages 794–803, 2011.

[43] C. Xiong, Z. Dai, J. Callan, Z. Liu, and R. Power. End-to-end neural ad-hoc ranking with kernel pooling. In *SIGIR'17*, pages 55–64, 2017.

[44] I. Yamada, H. Shindo, H. Takeda, and Y. Takefuji. Joint learning of the embedding of words and entities for named entity disambiguation. In *SIGNLL'16*, pages 250–259, 2016.

[45] B. Zhang and M. Al Hasan. Name disambiguation in anonymized graphs using network embedding. In *CIKM'17*, pages 1239–1248, 2017.

[46] C. Zhang, C. Huang, L. Yu, X. Zhang, and N. V. Chawla. Camel: Content-aware and meta-path augmented metric learning for author identification. In *WWW'18*, pages 709–718, 2018.

[47] Y. Zhang, F. Zhang, P. Yao, and J. Tang. Name disambiguation in aminer: Clustering, maintenance, and human in the loop. In *SIGKDD'18*, pages 1002–1011, 2018.

[48] X. Zhao. The scorecard solution to the author-paper identification challenge. In *KDD Cup 2013 Workshop*, page 4, 2013.

Bo Chen is a postgraduate student in Information School, Renmin University of China. His research interests include data integration and knowledge graph mining and he recently focuses on entity disambiguation in academic networks.

Jing Zhang received the master and PhD degree from the Department of Computer Science and Technology, Tsinghua University. She is an assistant professor in Information School, Renmin University of China. Her research interests include knowledge graph constructing and mining.

Jie Tang is a professor in the Department of Computer Science and Technology, Tsinghua University. His research interests include data mining, social network, and machine learning. He was honored with the SIGKDD Test-of-Time Award in Applied Data Science.

Lingfan Cai is a postgraduate student in Information School, Renmin University of China. His research interests includes name disambiguation in academic networks.

Zhaoyu Wang is a postgraduate student in the School of Computer Science and Technology, Anhui University. His research interests include name disambiguation in patent datasets.

Shu Zhao received her PhD degree from Anhui University in 2007. She is a professor in the School of Computer Science and Technology of Anhui University. Her current research interests include social network, machine learning and granular computing.

Hong Chen received PhD degree from the Institute of Computing Technology, CAS. She is a professor of Renmin University of China. Her research interests include data privacy, big data management, and data analysis based on new hardwares. She is a distinguished member of the CCF.

Cuiping Li received the PhD degree from the Institute of Computing Technology, CAS. She is currently a professor of Renmin University of China. Her current research interests include database systems, social network analysis, and data mining.