COAD: Contrastive Pre-training with Adversarial Fine-tuning for Zero-shot Expert Linking

Bo Chen, Jing Zhang*, Xiaokang Zhang, Xiaobin Tang,
Lingfan Cai, Hong Chen, Cuiping Li, Peng Zhang, Jie Tang, *Fellow, IEEE*

Abstract—Expert finding, a popular service provided by many online websites such as Expertise Finder, LinkedIn, and AMiner, benefits seeking consultants, collaborators, and candidate qualifications. However, its quality is suffered from a single source of support information for experts. This paper employs AMiner, a free online academic search and mining system, having collected more than over 100 million researcher profiles together with 200 million papers from multiple publication databases [36], as the basis for investigating the problem of expert linking, which aims at linking any external information of persons to experts in AMiner. A critical challenge is how to perform zero-shot expert linking without any labeled linkages from the external information to AMiner experts, as it is infeasible to acquire sufficient labels for arbitrary external sources. Inspired by the success of self-supervised learning in computer vision and natural language processing, we propose to train a self-supervised expert linking model, which is first pre-trained by contrastive learning on AMiner data to capture the common representation and matching patterns of experts across AMiner and external sources, and is then fine-tuned by adversarial learning on AMiner and the unlabeled external sources to improve the model transferability. Experimental results demonstrate that COAD significantly outperforms various baselines without contrastive learning of experts on two widely studied downstream tasks — author identification (improving up to 32.1% in HitRatio@1) and paper clustering (improving up to 14.8% in Pairwise-F1). Expert linking on two genres of external sources also indicates the superiority of the proposed adversarial fine-tuning method compared with other domain adaptation ways (improving up to 2.3% in HitRatio@1).

Index Terms—Expert Linking, Pre-training, Contrastive Learning, Adversarial Learning.

1 Introduction

Online websites such as Toptal[1], Expertise Finder[2], LinkedIn[3] and AMiner[4] provide valuable services of expert finding for governments or research groups to find consultants, collaborators and reviewers, and also for companies to find candidate qualifications, team dynamics, etc. However, the information about an expert is dispersed across different sources. For example, Google Scholar and AMiner mainly maintain the published papers, LinkedIn keeps the skills and background, while a large number of real-time news articles report the dynamic activities of experts. A single source of support information is far from comprehensive and convincing to support the high-quality expert finding, which demands to integrate heterogeneous support information about experts together.

In this paper, employing AMiner, a researcher-centric search system that allows users to find domain experts and collaborators, as the basis for our experimental data, we explain how to deal with the problem of expert linking across heterogeneous sources. AMiner is a free online academic search and mining system [36], which automatically collects researchers' profiles from the Web and integrates with published papers after name disambiguation [3], [50]. To date, it has collected more than 133,199,983 researchers' profiles together with 271,863,708 papers. We target at linking arbitrary information that describes an expert from external sources to the experts in AMiner. Figure 1 presents an example of the candidate AMiner experts for two ambiguous names mentioned in a news article. The problem is challenging as many candidates with the same affiliation or even similar research fields are to be distinguished.

Fig. 1. Linking news articles to AMiner experts. The candidate AMiner experts of two ambiguous names are presented.

Expert linking is closely related to the standard entity

- *Bo Chen, Jing Zhang, Xiaokang Zhang, Xiaobin Tang, Lingfan Cai, Hong Chen and Cuiping Li are with Information School, Renmin University of China, Beijing, China.*
  E-mail: allanchen224@gmail.com
  E-mail:{zhang-jing, zhang2718, txb, clfan, chong, licuiping}@ruc.edu.cn
  **Corresponding author*
- *Jie Tang, Peng Zhang are 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, zhangp31@tsinghua.edu.cn,

1. https://www.toptal.com/top-3-percent
2. https://expertisefinder.com/
3. http://www.linkedin.com
4. https://www.aminer.cn/

linking task, which usually links the mentions—the named entities extracted from the unstructured text—to the entities in a knowledge graph (often Wikipedia or Freebase) [7], [16], [19], [20], [42], [43]. The latest models usually convert the entities and the mentions into low-dimensional embeddings such that the aligned mentions and the entities can be close to each other. Most of the extensive study for the entity linking are in a supervised manner depending on a large labeled dataset such as AIDA CoNLL-YAGO and KBP Track at TAC 2010[5]. Unfortunately, the linkages between external information and the AMiner experts are often arduous to obtain. For example, in AMiner, it usually spends up to several hours to correct the collected papers for a top expert by a skilled annotator. Moreover, the external information about experts come from arbitrary sources, making it unforeseeable for us to annotate the corresponding labels beforehand. In view of this limitation of lacking the labels, we pay attention to the problem of zero-shot expert linking. A natural question arises: *can we learn a universal expert linking model from abundant AMiner experts such that it can be transferable to unseen external expert linking?*

The similar question has also be studied in natural language processing (NLP), computer vision (CV) and graph representation (GR). Facing abundant unlabeled data and a small number of labeled data, the most powerful paradigm in these fields is self-supervised learning targeting at pre-training a function on the large unlabeled data and fine-tuning it on the small number of labeled data. The function can represent a fundamental unit such as the natural language texts, images or graphs/nodes to capture the inherent correlations between them during pre-training, and is able to be transferred to represent the unseen units after fine-tuning. The typical self-supervised models include BERT [8] and XLNet [44] in NLP, MoCo [15] and SimCLR [4] in CV, and GCC [30] and GPT-GNN [18] in GR.

Present Work. Inspired by the success of pre-training in NLP, CV and GR, we propose to pre-train an expert linking model on the large unlabeled AMiner data. Because of the ultimate goal of expert linking, the model should be able to capture both the common representation patterns of experts and the matching patterns between experts across AMiner and external sources. To achieve this goal, two main challenges should be addressed. First, since an expert, usually described by different types of support information such as demographic attributes, papers and news articles, is neither a continuous, high-dimensional signal as an image nor a single discrete signal as a word, it is challenging to determine how to represent an expert and further how to measure the similarity of two experts. Second, sometimes, the gap of morphology, syntax, topics between the external sources and AMiner is obvious, which encourages us to fine-tune the pre-trained model on external sources to improve its transferability. But unlike the assumption of the small labeled data in downstream tasks, none of the labels can be obtained on most external sources, which increases the difficulty of fine-tuning.

We propose the COntrastive Pre-training with ADversarial Fine-tuning model (COAD) to enable expert linking from the external sources to AMiner experts. We leverage contrastive learning [39] to pre-train the model. The basic idea is to define expert discrimination as the pre-training task which samples, for each AMiner expert, the random instances, makes them from the same experts close and discriminates them from the different experts. Specifically, in COAD, we define an expert instance as a set of papers published by the expert, use BERT to encode each expert instance, and propose an interaction-based metric function to measure the similarity of two expert instances. Within the model, BERT can universally encode various types of support information such as demographic attributes, papers and news articles. And the interaction-based metric function can capture the fine-grained semantic matches between support information. After pre-training the BERT encoder and the metric function by contrastive learning, to make the functions transferable to unlabeled external sources, we leverage adversarial learning to align the representations of the external persons to those of the AMiner experts.

Although some pre-training models have been studied to tackle the zero-shot entity linking problem in some domains, the pre-training models are trained in a supervised manner on the large linkages of other domains [25]. However, the proposed pre-training model is in a self-supervised manner on the large unlabeled data.

We conduct extensive experiments on AMiner, News and LinkedIn to evaluate the performance of COAD. We evaluate the contrastive pre-training module and the adversarial fine-tuning module in COAD in two steps. Specifically, we first pre-train COAD on the pseudo labels of expert instances sampled from AMiner and evaluate its representation and the matching capacity by two intrinsic tasks on AMiner—author identification and paper clustering. After pre-training, we fine-tune COAD on both the AMiner data and the unlinked news articles or LinkedIn homepages. Then we evaluate the transferability of the model by the extrinsic task—linking from the name mentions in news articles or LinkedIn users to AMiner experts.

We summarize our contributions as follows:

- We propose COAD, consisting of a contrastive pre-training module and an adversarial fine-tuning module, to address zero-shot expert linking from external sources to AMiner.
- We design the pre-training task as expert discrimination to characterize the universal representation patterns of experts and matching patterns between experts. An interaction-based metric function is adopted to capture the fine-grained matches. Adversarial learning is leveraged to improve the transferability of pre-trained module.
- The extensive experiments on author identification and paper clustering of AMiner, and expert linking from news and LinkedIn to AMiner demonstrate the superior representation and matching capacity both within and across domains. All codes and data are publicly available[6].

2 Problem Formulation

This section defines the input of experts and formulate the problem of zero-shot expert linking in an self-supervised manner.

5. http://nlpprogress.com/english/entity_linking.html

6. https://github.com/BoChen-Daniel/Expert-Linking

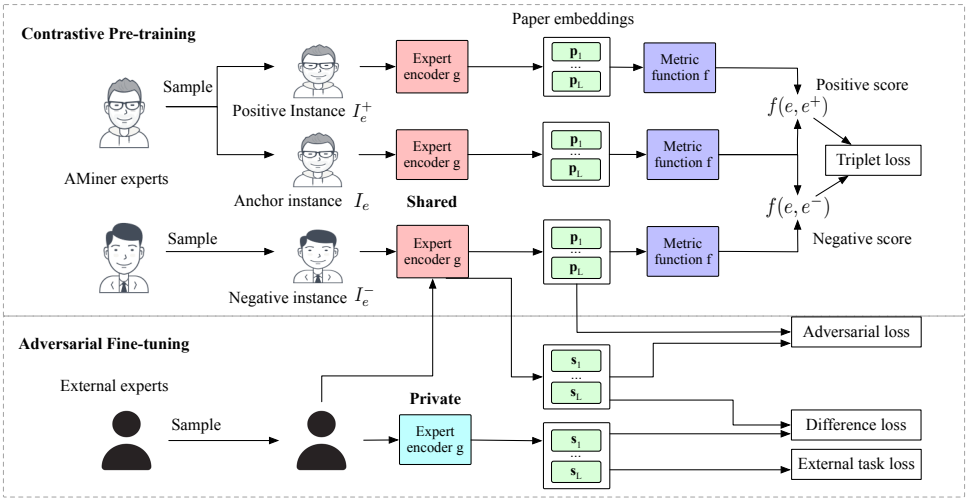


Fig. 2. **Model overview.** By discriminating the positive expert instance pair (I_e, I_e^+) from the negative instance pair (I_e, I_e^-), we train an expert encoder g and a metric function f, which are then fine-tuned on the unlabeled external experts by adversarial learning.

Definition 1. **Expert.** We define an expert e by a set of support information $c_e = \{s_1, s_2, \cdots, s_{n_e}\}$, where each s_i is a piece of support information to describe e. Notation n_e indicates the size of c_e.

The support information of experts differs from different sources. For example, if an expert e is derived from a news article, c_e can be the surrounding text of its name mentioned in the news article. The text can be divided into multiple sentences, indicating multiple pieces of support information. If e is derived from the LinkedIn website, c_e can be e's homepage, which contains multiple attributes such as summary, affiliations, skills, etc. Since we aim at linking arbitrary external information to AMiner, we particularly denote the support information of an AMiner expert by $c_e = \{p_1, \cdots, p_{n_e}\}$, where each p_i indicates a paper which contains title, keywords, venue, author names, author affiliations, etc.

As it is infeasible to prepare sufficient labeled data beforehand for training a supervised expert linking model on arbitrary external data, we address the zero-shot expert linking task in a self-supervised manner. Specifically, we study the problem of self-supervised expert linking by pre-training plus fine-tuning, one typical learning paradigm with self-supervision. The problem is to learn functions that can not only encode and align experts with similar support information on AMiner, but also be transferable to arbitrary external sources to encode and link unseen persons to AMiner experts. It is formulated as follows:

Problem 1. **Self-supervised Expert Linking.** Given a set of experts $\{e\}$ in AMiner, we aim at learning an expert encoder g that can represent each expert e into a low-dimensional feature vector. Based on the feature vector $g(e)$ for each epxert, we learn a metric function $f : \{g(e), g(e')\} \rightarrow \{y\}$ that can infer the alignment label y between any two experts e and e', where y is a binary value with $y = 1$ indicating e and e' are equivalent and $y = 0$ otherwise. After pre-training, we fine-tune g and f on $\{e\}$ and a set of external experts $\{\tilde{e}\}$ such that g and f are transferable to capture the representation patterns of $\tilde{e}$ and the matching patterns between $\tilde{e}$ and the corresponding expert e in AMiner.

As the problem is quite challenging, we assume any names from the external sources can be certainly aligned to an expert in AMiner. We leave the problem of non-existing alignment to the future work.

3 The COAD Framework

This section introduces the proposed self-supervised expert linking model consisting of two sub modules, the contrastive pre-training module and the adversarial fine-tuning module. The former module pre-trains the expert encoder g and the metric function f purely on AMiner by contrastive learning, and the later module fine-tunes g and f on both AMiner and the external data by adversarial learning. Figure 2 illustrates the whole framework of COAD.

3.1 Contrastive Pre-training Module

The pre-training module is to train an encoder to capture the universal representation patterns of experts. Based on the output of the encoder, it also learns a metric function to capture the universal matching patterns between experts. For this purpose, we need to design proper pre-training tasks. Instance discrimination in contrastive learning [39] is a widely used pre-training task in various domains. For example, in CV, random image instances are often obtained by croping, resizing or rotating of an image, and two image instances are contrasted according to the images from which they are created [4], [15]. Similarly, in NLP, word instances are contrasted according to whether they belong to the same context or not [8]. In graph representation, subgraphs are extracted from the neighbors of a node, and two sub graphs are contrasted according to their center nodes [30] .

Inspired by the above success of contrastive learning in CV, NLP and GR, we define expert discrimination as our pre-training task to perform contrastive learning. Via expert discrimination, we expect the output functions to be able to not only capture the representation patterns of experts but

also capture the matching patterns between two experts. To achieve this, four questions should be answered carefully:

- **Q1:** What is a random instance of an expert?
- **Q2:** How to encode an expert instance?
- **Q3:** What is an effective way to measure the similarity between two expert instances?
- **Q4:** Which kind of loss function should be selected?

Q1: A Random Instance of an Expert. According to the definition of expert in AMiner, we define a random instance of an expert as a set of randomly sampled papers of the expert. Specifically, given the support information $c_e = \{p_1, \cdots, p_{n_e}\}$ of an expert e, a random instance I_e of e can be formulated as follows:

$$I_e = \{p_1, \cdots, p_L\}, \tag{1}$$

where each $p_n \in c_e$ and L is the maximal number of papers sampled for each expert instance. Two random instances sampled from the same expert are viewed as a positive pair, while the two sampled from different experts are treated as a negative pair.

Q2: BERT-based Expert Encoder. BERT has advanced the state-of-the-art in various NLP tasks by pre-training on large Wikipedia dumps. Since the support information of the external experts may be in different languages from the papers of the experts in AMiner, we leverage a pre-trained multi-lingual BERT model [38], which projects words or sentences from different languages into the same semantic space, to encode the experts from different sources.

Straightforwardly, we can encode an expert by inputting all the tokens of its papers to BERT and taking the BERT output as the expert's representation. However, as many experts have a large number of papers, it is not feasible for BERT to accept so many tokens in all these papers simultaneously. In practice, we encode each paper as a basic representation unit of an expert instead of directly encoding each expert. Specifically, we concatenate the attributes of a paper, including title, keywords, venues, author names, author affiliations as the input of BERT and apply a Multi-layer Perceptron (MLP) on the BERT output as the paper embedding:

$$\mathbf{p} = g(p) = \text{MLP}(\text{CLS}(p)), \tag{2}$$

where $\text{CLS}(p)$ indicates the output CLS embedding of BERT. Then we take the embedding $\mathbf{p}$ of each paper as the input of the following metric function. Essentially, we connect the BERT model with the following metric function as an end-to-end framework to fine-tune BERT and learn the parameters of the metric function together.

Q3: Metric Function. We propose an interaction-based metric function to measure the similarity between two expert instances. Before that, we also explain the representation-based metric function and the reasons for the proposal.

Representation-based Metric Function. Representation-based metric function aims to encode each expert into an embedding, and then directly estimate the distance between two experts in the vector space. Specifically, we average the paper embeddings $\{\mathbf{p}_n\}_{n=1}^{L}$ of an expert instance I_e to obtain $\mathbf{e}$ and measure the Euclidean distance of $\mathbf{e}$ and $\mathbf{e}'$:

$$\mathbf{e} = \frac{1}{L}\sum_{n=1}^{L}\mathbf{p}_n, \qquad f(e, e') = \|\mathbf{e} - \mathbf{e}'\|^2. \tag{3}$$

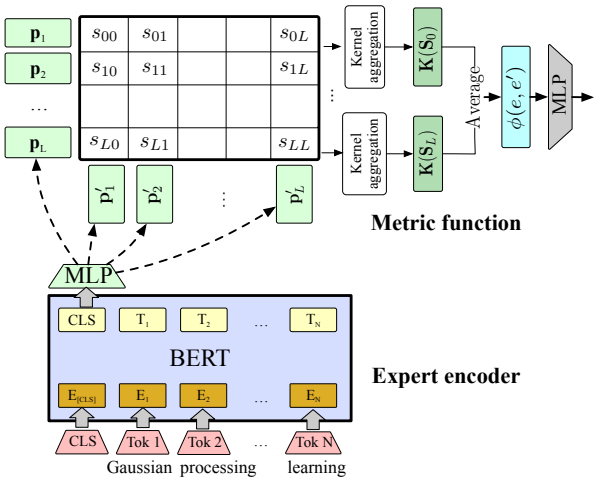


Fig. 3. The expert encoder and the metric function.

Interaction-based Metric Function. The representation-based metric function mixes an expert instance's papers together, which may suffer from semantic drift. Because if an expert publishes a large number of papers in multiple topics, two expert instances sampled from the same expert may include papers of different topics, but only a few similar papers. For this case, mixing all the paper embeddings of each expert instance will dilute the effect of the truly similar papers between two expert instances. To deal with the problem, we propose an interaction-based metric function to compare each pair of paper embeddings of two expert instances instead of comparing the embeddings of two expert instances. This similar idea is widely used in information retrieval to capture the exact and soft matches between a query and a candidate document [17], [40].

Formally, we utilize Eq.(2) to obtain a set of paper embeddings $\{\mathbf{p}_m\}_{m=1}^{L}$ and $\{\mathbf{p}'_n\}_{n=1}^{L}$ for I_e and $I_{e'}$ respectively, and then compute a similarity matrix $\mathbf{S}$ between the two embedding sets. Each element s_{mn} in $\mathbf{S}$ is computed by $s_{mn} = \|\mathbf{p}_m - \mathbf{p}_n\|^2$, standing for the Euclidean distance between the m-th paper and n-th paper in two expert instances. Then we apply an aggregation function to extract similarity patterns from the similarity matrix $\mathbf{S}$. As the papers of expert instances are order independent, we adopt a RBF kernel aggregation function [40], paying attention to how many similar pairs of papers within two expert instances, to extract the similarity patterns. Specifically, we first transfer each s_{mn} into $[0, 1)$ by the tanh function (Eq.(5)), and then transform it into a K-dimensional distribution (Eq.(7)), each k-th element of which is converted by the k-th RBF kernel with mean μ_k and variance σ_k (Eq.(6)), indicating how likely s_{mn} corresponds to the k-th similarity pattern. Then we sum up all the distributions over each row to represent the accumulated likelihood of the similarities between the m-th paper in l_e and all the papers in $I_{e'}$,

and further sum up all the rows into $\phi(e, e')$ to represent the similarity patterns between I_e and $I_{e'}$ (Eq.(8)). The log operation before the outermost summation in Eq.(8) is to reduce the negative influence of the different number of papers in different $I_{e'}$ for a given I_e.

$$s_{mn} = \|\mathbf{p}_m - \mathbf{p}_n\|^2, \quad (4)$$

$$\tilde{s}_{mn} = \tanh(s_{mn}), \quad (5)$$

$$K_k(\tilde{s}_{mn}) = \exp\left[-\frac{(\tilde{s}_{mn} - \mu_k)^2}{2\sigma_k^2}\right], \quad (6)$$

$$\mathbf{K}(\tilde{s}_{mn}) = [K_1(\tilde{s}_{mn}), \cdots, K_K(\tilde{s}_{mn})], \quad (7)$$

$$\phi(e, e') = \sum_{m=1}^{L} \log \sum_{n=1}^{L} \mathbf{K}(\tilde{s}_{mn}). \quad (8)$$

The kernel with $\mu = 0$ and $\sigma \to 0$ only considers the exact matches between two papers, but others such as $\mu = 0.5$, measure the semantic similarity between two papers. Finally, we apply a MLP layer

$$f(e, e') = \text{MLP}(\phi(e, e')) \quad (9)$$

to get the similarity between the expert instances I_e and $I_{e'}$.

Q4: Loss Function. We leverage the triplet loss instead of the widely used contrastive loss [15] in our problem. As mentioned by [50], the contrastive loss encourages different expert instances sampled from the same expert to be projected into a single point in the embedding space. Since many experts in AMiner publish papers on different topics, it is weird to force different papers into a single point. On the contrary, triplet loss maintains a distance between positive and negative pairs, without optimizing the absolute positions of the positive and negative pairs as the contrastive loss does. Thus the triplet loss is a better choice for contrastive learning over experts.

Specifically, for each anchor instance I_e sampled from the expert e, the positive counterpart I_e^+ is sampled from the same expert e, while a negative counterpart I_e^- is sampled from a different expert from e. I_e and I_e^+ comprise a positive pair. I_e and I_e^- comprise a negative pair. Given a set of triplets $\{(I_e, I_e^+, I_e^-)\}$, the triplet loss function is defined as:

$$\mathcal{L}^{\text{pre-train}}(\theta_g, \theta_f) = \sum_{(I_e, I_e^+, I_e^-)} \max\{0, m + f(e, e^-) - f(e, e^+)\}, \quad (10)$$

where m is a margin enforced between positive pairs and negative pairs, and θ_g, θ_f indicate the parameters of g and f respectively. Note to avoid trivial solutions, two instances within a positive pair are sampled without replacement to remove the overlap between the two instances.

3.2 Adversarial Fine-tuning Module

The pre-training module can be directly applied on the external sources for expert linking. Specifically, given an external expert $\tilde{e}$, we use the pre-trained expert encoder g to encode $\tilde{e}$ and apply the pre-trained metric function f to estimate the similarity between $\tilde{e}$ and each candidate expert e in AMiner. Then the top-matched expert can be viewed as the linking result of $\tilde{e}$. However, the morphology, syntax, topics of the external support information may be significantly different from the papers in AMiner, which encourages fine-tuning the pre-trained module to improve its transferability.

Algorithm 1: Training Process of COAD

Input: AMiner and external experts, learning rates μ_g, μ_f, μ_h.
Output: Learned parameters θ_g^{shared}, $\theta_g^{\text{private}}$, θ_f, θ_h.

```
/* The contrastive pre-training module.      */
repeat
    foreach minibatch of triplets {(I_e, I_e^+, I_e^-)} do
        θ_g^shared ← θ_g^shared − μ_g ∂L^pre-train/∂θ_g^shared; θ_f ← θ_f − μ_f ∂L^pre-train/∂θ_f;
until Converges;
/* The adversarial fine-tuning module.       */
repeat
    foreach minibatch of triplets {(I_e, I_e^+, I_e^-)} do
        θ_g^shared ← θ_g^shared − μ_g ∂L^pre-train/∂θ_g^shared; θ_f ← θ_f − μ_f ∂L^pre-train/∂θ_f;
    foreach minibatch of AMiner or external support information do
        θ_g^shared ← θ_g^shared + μ_g ∂L^adv/∂θ_g^shared; θ_h ← θ_h + μ_h ∂L^adv/∂θ_h;
    foreach minibatch of external support information do
        θ_g^shared ← θ_g^shared − μ_g ∂L^diff/∂θ_g^shared;
        θ_g^private ← θ_g^private − μ_g ∂L^diff/∂θ_g^private;
        θ_g^private ← θ_g^private − μ_g ∂L^ext/∂θ_g^private; θ_h ← θ_h − μ_h ∂L^ext/∂θ_h;
until Converges;
```

Recently, adversarial learning is widely adopted to achieve domain adaptation [2], [13], [24], [33]. The basic idea is, given a data sample, to design a generator to extract features for it, a classifier to predict the task-related label of it, and a domain discriminator to identify the source of it [13]. By trading-off the minimization of the task prediction loss and the maximization of the domain classification loss, the framework attempts to learn a generator that maps a sample from either the source or the target domain into a representation allowing the task predictor to classify source samples accurately, but crippling the ability of the domain classifier to distinguish the source of the sample.

We design an adversarial learning algorithm to fine-tune g and f to make them adapt to the external data. The specific generator, domain discriminator, task predictor and the corresponding loss functions are explained as follows.

Generator. We use the pre-trained expert encoder g as the shared generator to extract the similar features as the AMiner experts from the external experts. Meanwhile, we create a private generator by the same Eq.(2) to only extract the domain-specific features from the external experts. No private generator is executed on AMiner experts, as they are the targets to be aligned from the other external experts. To encourage the shared generator and private generator to encode different aspects of the external experts, following [24], we introduce orthogonality constraints as a difference loss:

$$\mathcal{L}^{\text{diff}}(\theta_g^{\text{shared}}, \theta_g^{\text{private}}) = \sum_{i=1}^{N_{\text{ext}}} ||g^{\text{shared}}(s_i)^{\text{T}} g^{\text{private}}(s_i)||_F^2, \quad (11)$$

where $||.||_F^2$ is the squared Frobenius norm, s_i is the i-th support information of the external source, N_{ext} is the

number of the pieces of the support information from the external source, g^{shared} is the shared generator with θ_g^{shared} as the parameters, and g^{private} and $\theta_g^{\text{private}}$ correspond to the shared generator.

Domain Discriminator. To detach the external private features from the shared feature space, we design a domain discriminator to distinguish the sources of the shared features. If the discriminator cannot distinguish the sources of the shared features, it indicates the features shared with the AMiner source are extracted from the external sources. Specifically, given a piece of support information s_i from either the AMiner source or the external source, we use the shared generator g^{shared} to extract its shared features, and apply a classifier h to predict whether it is from the external source or the AMiner source. We minimize the following loss function to confuse the classifier h via Gradient Reversed Layer [13], such that it cannot distinguish the source of any support information:

$$\begin{aligned}\mathcal{L}^{\text{adv}}(\theta_g^{\text{shared}}, \theta_h) &= \sum_{i=0}^{N_{\text{AMiner}}} \log(\hat{p}_i) + \sum_{i=0}^{N_{\text{ext}}} \log(1-\hat{p}_i), \qquad (12)\\ \hat{p}_i &= h\left(g^{\text{shared}}(s_i)\right) = \text{MLP}\left(g^{\text{shared}}(s_i)\right),\end{aligned}$$

where $\hat{p}_i$ indicates the probability of the support information s_i being from the AMiner source. N_{AMiner} refers to the number of the papers in AMiner. The classifier h is instantiated into a MLP layer.

Task Predictor. We define two task predictors. The first one is the same as the pre-training task (Cf. $\mathcal{L}^{\text{pre-traing}}$ in Eq.(10)) defined only on the AMiner source to guide the learning of the shared generator. The second one is defined only on the external source to predict whether the private features extracted by the private generator are from the external source or not. If the classifier can predict the source correctly, it indicates the private features are detached from the shared features. We minimize the following loss function to enhance the predictive ability of the classifier h, so that it can distinguish the source of any external private information:

$$\mathcal{L}^{\text{ext}}(\theta_g^{\text{private}}, \theta_h) = -\sum_{i=0}^{N_{\text{ext}}} \log(1-\hat{p}_i), \quad \hat{p}_i = h(g^{\text{private}}(\tilde{e}_i)), \qquad (13)$$

where $\hat{p}_i$ indicates the probability of the support information s_i encoded by g^{private} being from the AMiner source. The classifier h is the same as Eq.(12).

Overall Loss Function. We combine the loss functions by:

$$\mathcal{L}(\theta_g^{\text{shared}}, \theta_g^{\text{private}}, \theta_f, \theta_h) = \mathcal{L}^{\text{pre-train}} + \alpha\mathcal{L}^{\text{adv}} + \beta\mathcal{L}^{\text{diff}} + \gamma\mathcal{L}^{\text{ext}}, \quad (14)$$

where α, β and γ are trade-off hyper-parameters. Specifically, we first optimize Eq.(10) on AMiner experts to obtain an expert encoder g^{shared} and a metric function f by pre-training. Then by incorporating additional external experts, we optimize Eq.(14) to fine-tune g^{shared} and f by adversarial learning. Algorithm 1 presents the whole training process of the proposed COAD. The fine-tuned θ_g^{shared} and θ_f are used to represent and measure the similarity between AMiner and external experts.

Discussions. Most of the existing endeavors assume each domain is comprised of domain-agnostic features and domain-specific features, so they learn a shared generator and a private generator for each domain [24], [33]. However, in our problem, as the purpose is to link different external experts to a single source, i.e., AMiner experts, we need to extract the similar features as AMiner experts from external experts as much as possible, to allow the pre-trained metric function on AMiner to better capture the similarity patterns between external experts and AMiner experts. Thus we only create a private generator for the external experts to get rid of the dissimilar features with AMiner experts. The shared generator is to keep all the features of AMiner experts, but to extract partial features from the external experts. In addition, unlike the multi-task learning tasks [24], [33], the external sources are unlabeled, thus we introduce a novel domain prediction task purely on the external private features to better separate them from the shared features.

4 Experiments

In this section, we evaluate the proposed expert linking model on three datasets. We first evaluate the representation capacity of the pre-training module by the intrinsic tasks of author identification and paper clustering on AMiner. Then we evaluate the transferability of the fine-tuning module by the extrinsic task of external expert linking, i.e., linking external names from the news or the users from LinkedIn.com to AMiner experts.

4.1 Datasets.

Three datasets are introduced as below. Table 1 summarizes the data statistics.

- **AMiner [3].** We employ OAG-WhoIsWho[7], the largest name disambiguation dataset collected from AMiner, as the anchor dataset to be aligned. The dataset contains 421 names, 45,187 experts and 399,255 papers in English. The relationships between the papers and the experts are annotated by humans.
- **News.** We collect 20,658 news articles from several Chinese technique websites such as news.sciencenet.cn, jiqizhixin.com, etc. We extract 1,824 person names from these articles by a Chinese NER tool[8] and then link them to the experts in AMiner by a majority voting of three annotators' results. We extract six sentences before and after a name in a news article as the support information.
- **LinkedIn [49].** This dataset consists of 50,000 LinkedIn English homepages, 1,665 of which are linked to AMiner experts by human annotators via majority voting. We use affiliation, skills and summary in a homepage as the support information. The affiliation and the concatenated keywords in skills are separate pieces of support information. The long-text summary is divided into multiple pieces of support information.

7. https://www.aminer.cn/whoiswho
8. http://thulac.thunlp.org/

TABLE 1
Data statistics. Persons in AMiner are experts. Persons in News indicate the names mentioned in news articles. Persons in LinkedIn indicate the users in LinkedIn. The support information in them are papers, news articles and LinkedIn homepages respectively. #Average candidates is the average number of candidate AMiner experts for an author in a paper, for a name in a news article or for a LinkedIn user respectively.

	AMiner	News	LinkedIn
#Persons	45,187	1,824	1,665
#Support Information	399,255	20,658	50,000
#Average candidates	18	8.79	4.85

TABLE 2
Performance of Author Identification on AMiner.

Model	HR@1	HR@3	MRR
GBDT	0.873	0.981	0.927
Camel	0.577	0.737	0.644
HetNetE	0.582	0.759	0.697
CONNA	**0.911**	**0.985**	**0.949**
Unsupervised	0.713	0.875	0.808
Paper-paper pseudo labels	0.864	0.960	0.915
Paper-expert pseudo lables	0.892	0.970	0.933
Representation metric function	0.870	0.956	0.918
COAD (pre-train)	0.898	0.964	0.934

Name Variants. Given a person to be linked, the candidates have variant names obtained by moving the last name to the first or 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". The queried names in the News articles are translated into English before linking. Advanced blocking techniques [1], [34] can be considered in the future to improve the quality of the candidates.

4.2 Evaluation of the Pre-training Module.

We adopt two widely-studied tasks, author identification [3], [5], [47] and paper clustering [26], [46], [50], as the intrinsic tasks to directly evaluate the representation capacity of the pre-training module without fine-tuning. Both of the evaluations are performed only on AMiner. This section explains the experimental settings and results on these two tasks. More details about the pre-training and test settings are left in the Appendix.

4.2.1 Author Identification

Task Description. Author identification follows the second task of a name disambiguation competition[9] to assign a new paper to the right author in AMiner. Thus, we adopt the same test set as the competition. After pre-training, we estimate the similarity score between a new paper and each candidate expert in the test set by the pre-trained encoder g and the metric function f. Then we rank all the candidate experts by the similarity scores and return the top-ranked expert as the right answer.

Evaluation Metrics. We use HitRatio@K (HR@K, K=1,3) to measure the proportion of the correctly assigned experts ranked in top K candidates, and use MRR to measure the average reciprocal ranks of the correctly assigned experts.

Baselines. The standard entity linking task links the mentions extracted from the unstructured text to the entities in Wikipedia or Freebase [31]. Compared with it, the mentions and entities of author identification refer to papers and experts, which are more complex objects, making it infeasible to directly apply the standard entity linking methods to identify experts for papers. The baselines addressing author identification specially are as follows:

- ***GBDT*** [11], [23]: is a frequently adopted model to address the same problem in KDD Cup 2013 [32]. We train a GBDT model with hand-crafted features shown in Table 6 to measure the similarity between a paper and an expert.
- ***Camel*** [47]: represents a paper by GRU-embeded title and keywords and represents an expert by an one-hot embedding. The metric function is the Euclidean distance between the embeddings of a paper and an expert. Camel is optimized by the same triplet loss as Eq. (10). To enable Camel to align unseen experts in the test set, we represent each unseen expert by averaging its paper embeddings.
- ***HetNetE*** [5]: is similar to Camel except that each paper is represented by the author names, affiliations, venues in addition to the title and keywords.
- ***CONNA*** [3]: is also an interaction-based model, but the basic interaction matrix is built between the token embeddings of two attributes such as titles or authors. Then the matrices of different attributes are aggregated as the interactions between the new paper and a paper in an expert, finally different papers interactions are aggregated as the interactions between the new paper and an expert.

Experimental Results. Table 2 shows the performance of the pre-training module evaluated by author identification on AMiner. It is averaged over 30 runs with random initailizations of model parameters.

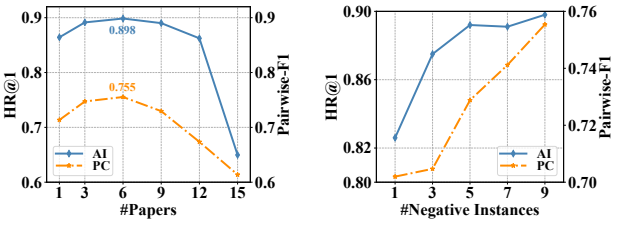
From the results, we can see that COAD (pre-train) outperforms Camel and HetNetE by 31.6∼32.1% in HR@1. Instead of the interaction-based metric function in COAD (pre-train), Camel and HetNetE adopt a representation-based metric function, which embed a paper and an expert respectively, and compute a single similarity score between them. They fail to capture the fine-grained similarities between papers and experts. Besides, one-hot embeddings for experts aren't able to address unseen experts. Although we avoid the issue by averaging the paper embeddings of an expert, the testing deviates from the training process.

COAD (pre-train) is comparable to GBDT and CONNA. GBDT, extracting the interaction features between a paper and an expert, and CONNA, computing the interactions of the tokens between a paper and an expert, perform much better than the representation-based Camel and HetNetE (+29.1∼33.4% in HR@1), which also demonstrates the effectiveness of the interaction-based strategy. CONNA even outperforms COAD (pre-train) by 1.3% in HR@1, because CONNA is more carefully designed, which first builds the basic interactions of the tokens between two attributes and then aggregates the similarities of attributes and the similarities of the papers together by attention mechanisms. How-

9. https://biendata.xyz/competition/aminer2019_2/

TABLE 3
Performance of Paper Clustering on AMiner.

Model	P-Pre.	P-Rec.	P-F1
Louppe et al.	0.609	0.605	0.607
Zhang et al.	0.768	0.551	0.642
G/L-Emb	**0.835**	0.640	0.724
Unsupervised	0.332	0.591	0.425
Paper-paper pseudo labels	0.659	0.779	0.714
Paper-expert pseudo lables	0.715	0.786	0.749
Representation metric function	0.595	0.754	0.665
COAD (pre-train)	0.724	**0.789**	**0.755**



(a) Paper Sampling. (b) Negative Sampling.

Fig. 4. The Effect of (a) paper sampling and (b) negative sampling. AI and PC stand for author identification and paper clustering respectively.

ever, GBDT and CONNA, the models specially designed for author identification on AMiner, cannot be adapted to other domains. Instead of directly computing the interactions between tokens, COAD (pre-train) first encodes each paper into a basic embedding and then computes the interactions between papers and experts. This expert encoder g can be generalized to other domains such as news articles to encode sentences or LinkedIn homepages to encode users' attributes. Moreover, instead of optimizing the relationship between a paper and an expert by GBDT and CONNA, COAD (pre-train) performs a contrastive learning between two expert instances, which can result in a more general encoder g to benefit multiple downstream tasks.

4.2.2 Paper Clustering

Task Description. Paper clustering, aiming at clustering papers belonging to a same person together, is a classical yet important task on online academic platforms. Since it is exactly the first task of the name disambiguation competition[10], we adopt the same test set as the competition. We cluster all the papers with a same author name in the test set by hierarchical agglomerative clustering algorithm (HAC) based on the paper embeddings generated by the expert encoder g.

Evaluation Metrics. We use pairwise Precision, Recall, and F1-score (P-Pre., P-Rec, and P-F1) to evaluate the clustering results of each name, and then calculate the macro averaged score of each metric over all the names.

Baselines. We compare with three state-of-the-art methods for paper clustering. They all use HAC to cluster papers. For a fair comparison, we assume the true number of clusters within each name is known. The differences lie in the embedding mechanisms.

- ***Louppe et al.*** [26]: trains a similarity function based on hand-crafted pairwise features to directly measure the similarity between papers without embedding them.
- ***Zhang et al.*** [46]: constructs an expert-expert network (two coauthors are linked), an expert-paper network (a paper is linked to its author) and a paper-paper network (two papers coauthored by the same author are linked) for each name. Then they perform graph embedding by preserving node connectivity on each network to learn paper embeddings.
- ***G/L-Emb*** [50]: first learns paper embeddings on a global paper-paper network (two papers coauthored by the same author are linked) by graph embedding and then fine-tunes the paper embeddings on a local paper-paper network built for each name (two similar papers are linked) by graph auto-encoding.

10. https://biendata.xyz/competition/aminer2019/

Experimental Results. Table 3 presents the performance of the pre-training module evaluated by paper clustering on AMiner. Overall, COAD (pre-train) advances other baselines by 3.1∼14.8% in terms of Pairwise-F1.

Among all the baselines, the model proposed by Louppe et al. performs the worst. Because the global distribution of the papers, largely determined by the embeddings of the papers, plays the critical role on the clustering results. However, instead of learning paper embeddings, this baseline learns the local similarity of two papers, which is more suitable to the classification task such as author identification. the model proposed by Zhang et al. and G/L-Emb, building the local paper-paper network for each name and leveraging the graph structures to learn paper embeddings, outperform the model proposed by Louppe et al. by 3.5∼11.7% in Pairwise-F1. In addition to the local embedding, G/L-Emb learns global embeddings via preserving the connectivity in global networks, making it outperform the model proposed by Zhang et al by +8.2% in Pairwise-F1. Our model, COAD (pre-train), discards the local graph embedding for its limitation in inductive learning. But COAD (pre-train) still performs the best because of the contrastive learning between expert instances, which results in a general encoder g to be adapted to various downstream tasks.

4.3 Analysis of the Pre-training Module.

Ablation Study. To verify the efficacy of different components in COAD (pre-train), we make four variants:

- ***Unsupervised:*** directly uses the pre-trained multi-lingual BERT to obtain a CLS embedding for each paper, and averages all the paper embeddings for an expert. Euclidean distance is to measure the similarity between two papers or between a paper and an expert.
- ***Paper-paper pseudo labels:*** samples paper-paper pseudo labels instead of the pairs of two expert instances, where two papers are viewed as a positive pair if they are published by a same expert and a negative pair otherwise.
- ***Paper-expert pseudo labels:*** samples paper-expert pseudo labels instead of two expert instances, where a paper-expert pair is a positive pair if the paper is published by the expert and a negative pair otherwise.

- ***Representation metric function:*** uses the representation-based metric function formulated by Eq.(3) instead of the interaction metric function in our module.

We present the performance of the above variant models in Table 2 and Table 3. First, we can see that the unsupervised model performs the worst (-18.5% in HR@1 and -33.0% in Pairwise-F1 compared with COAD (pre-train)), which indicates that BERT without any fine-tuning fails to measure the inherent correlations between papers or experts. Second, both the paper-paper pseudo labels and the paper-expert pseudo labels underperform our module by 0.6∼3.4% in HR@1 and by 0.6∼4.1% in Pairwise-F1, which indicates that contrasting two expert instances by COAD (pre-train) can result in better representations of papers and experts. Finally, the representation-based metric function underperforms our module by 2.8% in HR@1 and by 9.0% in Pairwise-F1, which emphasizes the advantage of the interaction-based metric function.

Effect of Paper Sampling. We investigate how the maximal number L of the papers sampled for an expert instance affects the performance of COAD (pre-train). We vary L (#Papers) from 1 to 15 with interval 3 and present the performance of both author identification and paper clustering in Figure 4(a). We can see that when #Papers equals to 6, COAD (pre-train) performs the best in both the tasks. Either too few or two many papers sampled for an expert will harm the performance. Too few papers may result in particularly dissimilar expert instances even if they are sampled from a same expert, making it difficult to distinguish the positive pairs of expert instances from the negative pairs. On the contrary, two many papers will result in particularly close expert instances if they are sampled from a same expert. The loss of these positive pairs will become zero, leading to homogeneously pushing all the expert instances away from each other, which is non-intuitive to justify [45].

Effect of Negative Sampling. The number of negative instances (#Negative instances) also influences the model performance. Many endeavors [4], [15] have shown that more negative instances can result in better performance. To verify the conclusion, we vary the number of the negative instances from 1 to 9 with interval 2 in both the tasks and present the results in Figure 4(b). We can see that when the number of the negative instances increases from 1 to 9, COAD (pre-train) consistently improves HR@1 by 7.2% on author identification and improves Pairwise-F1 by 5.3% on paper clustering. Since all the experiments are conducted on a single NVIDIA Tesla P40 with 24 GB memory, more than 9 negative instances for each positive instance will exceed the capacity of GPUs even if the batch size is set to 1. Thus we set the number of the negative instances as 9. But it is possible to improve the performance when GPUs with larger memory size are available.

4.4 Evaluation of the Adversarial Fine-tuning Module.

We also evaluate the transferable ability of our model by external expert linking. We fine-tune COAD (pre-train) on both AMiner and the unlabeled external dataset, and evaluate it on the test set. Specifically, we represent each external person and AMiner expert by the fine-tuned encoder g and measure the similarity of them by the fine-tuned metric function f. HR@K and MRR are also used as the metrics to evaluate the ranked AMiner candidates.

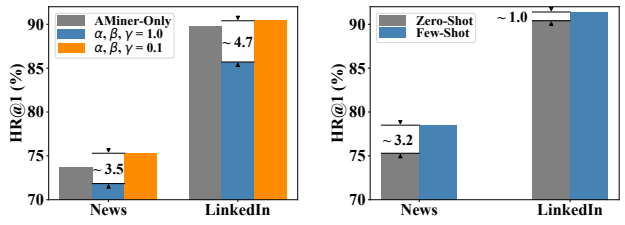
TABLE 4
Performance of External Expert Linking to AMiner.

External Sources	News			LinkedIn		
	HR@1	HR@3	MRR	HR@1	HR@3	MRR
Chain Pre-training	0.739	0.927	0.839	0.895	0.978	0.939
DANN	0.743	0.928	0.842	0.901	0.983	0.943
ASP-MTL	0.746	0.930	0.843	0.903	0.983	0.944
Unsupervised	0.329	0.731	0.559	0.805	0.963	0.886
AMiner-Only	0.737	0.927	0.837	0.897	0.982	0.940
COAD	**0.753**	**0.936**	**0.848**	**0.904**	**0.987**	**0.945**
COAD w/o $\mathcal{L}^{\text{adv}}$	0.721	0.923	0.825	0.873	0.981	0.927
COAD w/o $\mathcal{L}^{\text{diff}}$	0.745	0.927	0.841	0.901	0.985	0.942
COAD w/o $\mathcal{L}^{\text{ext}}$	0.743	0.925	0.84	0.898	0.983	0.941

Baselines. We compare with three domain adaptation methods which transfer the encoder g and the metric function f from AMiner to the external data.

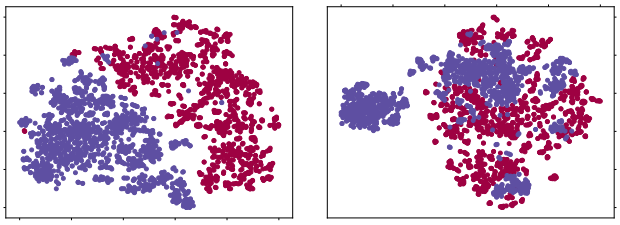
- ***Chain Pre-training*** [25]: chains together a series of pre-training stages for domain adaptation. Following it, we first fine-tune the original BERT on both AMiner and the external data, then fine-tune it only on the external data, and finally train COAD (pre-train) based on the fined-tuned BERT on AMiner data.
- ***DANN*** [13]: extracts domain-agnostic but not domain-private features. Following it, we only use a shared generator to encode both AMiner and external experts.
- ***ASP-MTL*** [24]: extracts both the domain-shared and domain-private features. Following it, in addition to a shared generator, we create both AMiner private generator and the external private generator.

Experimental Results. Table 4 shows the performance of linking from the external persons mentioned in news articles or from the LinkedIn users to AMiner experts. All the results are averaged over 30 runs with random initializations of the model parameters. We can see that the proposed COAD outperforms all the baselines by 0.1∼1.4% in terms of HR@1. The Chain Pre-training baseline performs the worst, as it fine-tunes only the BERT encoder g but not the metric function f. Although DANN fine-tunes both g and f, it does not detach the private features from the shared features. On the contrary, ASP-MTL, creating both the private and shared encoders, outperforms the above two baselines. Compared with ASP-MTL, the proposed COAD creates the private encoder only for the external data but not for AMiner data. Because the aim of external expert linking is to extract the similar features as AMiner experts from the external persons as much as possible, there is no need to extract the private features of AMiner experts. Moreover, we create a special loss function to predict whether the private features are from the external data or not, which makes the external private features special enough to be identified.



(a) Hyper-parameters. (b) Labels.

Fig. 5. The effect of the hyper-parameters and the labeled data in the external sources.



(a) Before fine-tuning. (b) After fine-tuning.

Fig. 6. T-SNE visualization before and after the adversarial fine-tuning for news articles (*Purple*) and AMiner papers (*Red*).

4.5 Analysis of the Fine-tuning Module.

Ablation Study. To verify different components of the fine-tuning module, five model variants are also compared.

- ***Unsupervised:*** is the same as the unsupervised variant model for evaluating the pre-training module.
- ***AMiner-Only***: directly applies COAD (pre-train) to encode and link the external persons to AMiner experts.
- ***COAD w/o*** $\mathcal{L}^{adv}$*:* removes the domain discriminator in Eq.(12) from COAD.
- ***COAD w/o*** $\mathcal{L}^{diff}$*:* removes the orthogonality constraints in Eq.(11) from COAD.
- ***COAD w/o*** $\mathcal{L}^{ext}$*:* removes the external private predictor in Eq.(13) from COAD.

We present the performance of the above variant models in Table 4. Among them, the unsupervised model performs the worst (-9.9∼42.4% in HR@1 compared with COAD), as the original BERT isn't fine-tuned at all on the expert linking data. Although the BERT encoder g and the metric function f in AMiner-Only are fine-tuned on AMiner, they aren't fine-tuned on the external data at all, thus AMiner-Only still underperforms COAD by 0.7∼1.6%.

Among the three model variants which remove different loss functions from COAD , COAD w/o $\mathcal{L}^{\text{adv}}$ performs the worst (-3.1∼3.2% in HR@1 compared with COAD), which indicates the domain discriminator takes the most important role in the fine-tuning process. Both COAD w/o $\mathcal{L}^{\text{diff}}$ and COAD w/o $\mathcal{L}^{\text{ext}}$ underperform COAD by 0.3∼1.0% in HR@1, which demonstrates that both the loss functions in our adversarial fine-tuning module can enhance the model performance. We also observe that COAD w/o $\mathcal{L}^{\text{diff}}$ performs slightly better than COAD w/o $\mathcal{L}^{\text{ext}}$ by 0.2∼0.3% in HR@1. Because we expect more about distinguishing the external private features from the AMiner features than forcing the private and the shared features of the external data orthogonal with each other.

Effect of Hyper-parameters. We investigate whether the values of α, β and γ, the weights to balance the four loss functions in Eq.(14), will affect the fine-tuning performance. We compare three settings, $\alpha, \beta, \gamma = 0$ (i.e., AMiner-Only), $\alpha, \beta, \gamma = 1$ and $\alpha, \beta, \gamma = 0.1$, and present the results in Figure 5(a). It is shown that the larger weights for adversarial learning harms the fine-tuning performance, which is even worse than the pure pre-training model AMiner-Only. Because with larger adversarial weights, the shared generator in the fine-tuning module will be trained more thoroughly. The representations of the AMiner experts will be closer to those of the external persons. However, the metric function, being trained only on the pseudo labels of AMiner, will be harmed if it accepts the more external-person-like experts as input.

Effect of Labels in the External Sources. We investigate whether the model performance will be improved when adding additional labeled data from the external sources. Specifically, we add 202 linkages from the persons mentioned in news articles to AMiner experts, and add 336 linkages from the LinkedIn users to AMiner linkages when fine-tuning our model. The prediction loss on these additional labels is the same as Eq.(10). The results in Figure 5(b) show that a few labels from the external sources can indeed improve the external linking performance. Moreover, we observe that the performance growth is more obvious on news than on LinkedIn. Because the news articles are in different language from AMiner data. It is more difficult for the expert encoder and the metric function to be transferred to the external data without any labeled data. Thus the performance growth is more significant when adding labels on news data.

Visualization. In Figure 6, we utilize t-SNE [27] to visualize the papers from AMiner and the sentences from the news articles embedded by the expert encoder g trained before and after the adversarial fine-tuning. Figure 6(a) indicates that when applying COAD (pre-train), the embedding distribution of the news articles is not aligned with that of the AMiner papers, while Figure 6(b) shows that COAD , after adversarial fine-tuning, can mitigate the domain shift.

4.6 Online Deployment

COAD has been deployed online for linking news articles to AMiner experts[11]. In the demo, each name extracted from a news article is linked to the corresponding AMiner expert with a brief introduction on the right of the news article. Due to the efficiency of the BERT model, the linkages from the news articles to AMiner experts are predicted by COAD offline. The efficiency issue will be addressed in future works. Since a name mention may not be linked to any AMiner experts, we train an additional classifier using the output similarity embeddings in Eq.(8) as features to predict whether the mention should be linked to the top-ranked expert by COAD or not [3].

11. https://newsminer.net/ExpertLinking/

5 Related Work

Entity Linking. Most of the works on entity linking focus on a standard setting which links the mentions from the unstructured text to the entities in a knowledge graph (often Wikipedia of Freebase) [7], [16], [19], [20], [42], [43], where the mentions and entities are usually represented by CNN [28], LSTM [14], bag-of-words embeddings [12], etc. Recent work has also investigated a cross-domain setting, which links mentions from different types of text such as blogposts and news articles to Wikipedia. They train a model only on the labeled mentions in Wiki-pages and directly apply it on different text [14], [22]. However, the gap between the support information of AMiner experts and the external persons is much more significant than that between Wiki-pages and other text, which prevents us from directly using the pre-trained model on AMiner to any external sources. Logeswaran et al. [25] trains a BERT-based mention-entity linking model on the source domain and then adapt it to the target domain. Our work differs from it in two aspects: first, they have explicit labels on the source domain, while the linkage labels are unknown on AMiner; second, an expert comprised by multiple papers is more complex than an entity represented by a text description.

The problem is also related to Entity Alignment, which targets at aligning the entities coming from different knowledge graphs but indicating the equivalent ones. Factor graph model [49], TransE [35], graph convolution networks [41], [48] and BERT-based interaction matching model [37] are studied to solve the problem, but most of them are in a supervised way.

Contrastive Learning. Contrastive learning is widely used to pre-train the representations of images or words, where an instance of an image is usually formed by transforming images [10], ordering patches [9] or segmenting objects [29], and the pseudo-labels for words are usually formed by the contextual words or sentences [8], [44]. Different from them, in addition to learn expert representations, we also need to train a good metric function for matching experts by contrastive learning.

The proposed pre-training task of expert discrimination is related to name disambiguation, which aims to partition the given papers into a set of disjoint clusters to make each cluster correspond to a real world person [46], [50]. Expert discrimination is also related to the problem of author identification, which is to assign the authors to an anonymous paper [5], [47]. Both the tasks learn paper or author embeddings based on the ownership of papers to authors. Different from them, we follow the idea of contrastive learning to contrasting random expert instances.

Adversarial Domain Adaptation. Adversarial learning has been extensively studied for cross-lingual and cross-domain transfer [13] such as word translation [6], [21], text classification [24] and relation extraction [33] in two different domains. Different from the existing end-to-end adversarial learning, we adopt adversarial learning for fine-tuning, which is an independent process following a pre-training stage.

6 Conclusions

We solve that problem of zero-shot expert linking from external sources to AMiner. We propose COAD consisting of a pre-training module and a fine-tuning module. The former one performs the pre-training task of expert discrimination by contrastive learning, which learns an expert encoder to capture the universal representation patters of experts and learns a metric function to capture the universal matching patterns between experts. The later one fine-tunes the functions on both AMiner and the external source to improve their transferability. Experimental results on AMiner and two external sources, news articles and LinkedIn homepages, demonstrate the superiority of the proposed model compared with the state-of-the-art baselines and variants. In the future, NIL recognition will be considered. It is also promising to design a better encoder than BERT to encode the structured and semi-structured support information of an expert.

References

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

[2] P. Cao, Y. Chen, K. Liu, and J. Zhao. Adversarial transfer learning for chinese named entity recognition with self-attention mechanism. In *EMNLP'18*, pages 182–192, 2018.

[3] B. Chen, J. Zhang, J. Tang, L. Cai, Z. Wang, S. Zhao, H. Chen, and C. Li. Conna: Addressing name disambiguation on the fly. *TKDE'20*, 2020.

[4] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

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

[6] X. Chen and C. Cardie. Unsupervised multilingual word embeddings. In *EMNLP'18*, pages 261–270, 2018.

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

[8] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[9] C. Doersch, A. Gupta, and A. A. Efros. Unsupervised visual representation learning by context prediction. In *ICCV'15*, pages 1422–1430, 2015.

[10] A. Dosovitskiy, J. T. Springenberg, M. Riedmiller, and T. Brox. Discriminative unsupervised feature learning with convolutional neural networks. In *NIPS'14*, pages 766–774, 2014.

[11] 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.

[12] O.-E. Ganea and T. Hofmann. Deep joint entity disambiguation with local neural attention. In *EMNLP'17*, 2017.

[13] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. Domain-adversarial training of neural networks. *JMLR*, pages 2096–2030, 2016.

[14] N. Gupta, S. Singh, and D. Roth. Entity linking via joint encoding of types, descriptions, and context. In *EMNLP'17*, pages 2681–2690, 2017.

[15] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum contrast for unsupervised visual representation learning. *arXiv preprint arXiv:1911.05722*, 2019.

[16] F. Hou, R. Wang, J. He, and Y. Zhou. Improving entity linking through semantic reinforced entity embeddings. In *ACL'20*, pages 6843–6848, 2020.

[17] 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.

[18] Z. Hu, Y. Dong, K. Wang, K.-W. Chang, and Y. Sun. Gpt-gnn: Generative pre-training of graph neural networks. In *KDD'20*, pages 1857–1867, 2020.

[19] J.-C. Klie, R. E. de Castilho, and I. Gurevych. From zero to hero: Human-in-the-loop entity linking in low resource domains. In *ACL'20*, pages 6982–6993, 2020.
[20] N. Kolitsas and O.-E. Ganea. End-to-end neural entity linking. In *ACL'18*, 2018.
[21] G. Lample, A. Conneau, M. Ranzato, L. Denoyer, and H. Jegou. Word translation without parallel data. In *ICLR'18*, 2018.
[22] P. Le and I. Titov. Improving entity linking by modeling latent relations between mentions. In *ACL'18*, 2018.
[23] 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.
[24] P. Liu, X. Qiu, and X. Huang. Adversarial multi-task learning for text classification. In *ACL'17*, pages 1–10, 2017.
[25] L. Logeswaran, M.-W. Chang, K. Lee, K. Toutanova, J. Devlin, and H. Lee. Zero-shot entity linking by reading entity descriptions. In *ACL'19*, 2019.
[26] G. Louppe, H. T. Al-Natsheh, M. Susik, and E. J. Maguire. Ethnicity sensitive author disambiguation using semi-supervised learning. In *international conference on knowledge engineering and the semantic web*, pages 272–287. Springer, 2016.
[27] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *JMLR*, 9(Nov):2579–2605, 2008.
[28] S. Murty, P. Verga, L. Vilnis, I. Radovanovic, and A. McCallum. Hierarchical losses and new resources for fine-grained entity typing and linking. In *ACL'18*, pages 97–109, 2018.
[29] D. Pathak, R. Girshick, P. Dollár, T. Darrell, and B. Hariharan. Learning features by watching objects move. In *CVPR'17*, pages 2701–2710, 2017.
[30] J. Qiu, Q. Chen, Y. Dong, J. Zhang, H. Yang, M. Ding, K. Wang, and J. Tang. Gcc: Graph contrastive coding for graph neural network pre-training. In *KDD'20*, pages 1150–1160, 2020.
[31] D. Roth, H. Ji, M.-W. Chang, and T. Cassidy. Wikification and beyond: The challenges of entity and concept grounding. In *ACL (Tutorial Abstracts)*, page 7, 2014.
[32] 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.
[33] G. Shi, C. Feng, L. Huang, B. Zhang, H. Ji, L. Liao, and H. Huang. Genre separation network with adversarial training for cross-genre relation extraction. In *EMNLP'18*, pages 1018–1023, 2018.
[34] R. C. Steorts, S. L. Ventura, M. Sadinle, and S. E. Fienberg. A comparison of blocking methods for record linkage. In *PSD'14*, pages 253–268, 2014.
[35] Z. Sun, W. Hu, Q. Zhang, and Y. Qu. Bootstrapping entity alignment with knowledge graph embedding. In *IJCAI'18*, pages 4396–4402, 2018.
[36] 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.
[37] X. Tang, J. Zhang, B. Chen, Y. Yang, H. Chen, and C. Li. Bert-int:a bert-based interaction model for knowledge graph alignment. In *IJCAI-20*, pages 3174–3180, 2020.
[38] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush. Transformers: State-of-the-art natural language processing. In *EMNLP'20: System Demonstrations*, pages 38–45, Oct. 2020.
[39] Z. Wu, Y. Xiong, S. X. Yu, and D. Lin. Unsupervised feature learning via non-parametric instance discrimination. In *CVPR'18*, 2018.
[40] 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.
[41] K. Xu, L. Wang, M. Yu, Y. Feng, Y. Song, Z. Wang, and D. Yu. Cross-lingual knowledge graph alignment via graph matching neural network. In *ACL'19*, pages 1452–1461, 2019.
[42] I. Yamada, A. Asai, H. Shindo, H. Takeda, and Y. Matsumoto. Luke: Deep contextualized entity representations with entity-aware self-attention. In *EMNLP'20*, pages 6442–6454, 2020.
[43] 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.
[44] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *NIPS'19*, pages 5754–5764, 2019.
[45] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen. Graph contrastive learning with augmentations. *NIPS'20*, 33, 2020.
[46] B. Zhang and M. Al Hasan. Name disambiguation in anonymized graphs using network embedding. In *CIKM'17*, pages 1239–1248, 2017.
[47] 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.
[48] J. Zhang, B. Chen, X. Wang, H. Chen, C. Li, F. Jin, G. Song, and Y. Zhang. Mego2vec: Embedding matched ego networks for user alignment across social networks. In *CIKM'18t*, pages 327–336, 2018.
[49] Y. Zhang, J. Tang, Z. Yang, J. Pei, and P. S. Yu. Cosnet: Connecting heterogeneous social networks with local and global consistency. In *SIGKDD'15*, pages 1485–1494, 2015.
[50] 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.

Bo Chen is a graduate 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 associate professor in Information School, Renmin University of China. Her research interests include knowledge graph constructing and mining.

Xiaokang Zhang is an undergraduate student in Information School, Renmin University of China. His research interests includes knowledge graph mining.

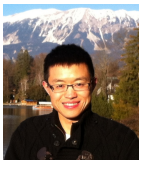
Lingfan Cai is a graduate student in Information School, Renmin University of China. His research interests includes name disambiguation in academic networks.

Xiaobin Tang is a postgraduate student in Information School, Renmin University of China. His research interests includes knowledge graph and cross-lingual entity alignment.

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.

Peng Zhang is a senior engineer in the Department of Computer Science and Technology, Tsinghua University, and also a Ph.D. of Tsinghua University Innovation Leadership Project. He focused in text mining, knowledge graph construction and application.



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.

APPENDIX A

TABLE 5
Hyper-parameters of COAD.

Model architecture	
BERT MLP	$\mathbb{R}^{768\times512}$
#Kernels K	21
Metric MLP	$\mathbb{R}^{21\times21}$, $\mathbb{R}^{21\times1}$
Margin m	1.0
Adversarial MLP	$\mathbb{R}^{512\times100}$, $\mathbb{R}^{100\times2}$
LeakyReLU	Negative slope is 0.2
Learning rate	
μ_g for encoder	2e-5
μ_f for metric function	2e-3
μ_h for discriminator	1e-3
Learning rate decay	Exponential decay = 0.96
Loss function weight	
α for adversarial loss	0.1
β for difference loss	0.1
γ for external task loss	0.1
Batch size	
For pre-training on AMiner data	32
For fine-tuning on External sources	256
Others	
#Negative Instances	9
#Papers	6
Optimization algorithm	Adam
Pre-training epochs	20
Fine-tuning epochs	1

A.1 Implementation Details

The detailed hyper-parameters are listed in Table 5.

A.1.1 Running Environment

We implement the model by Python 3.6.8, PyTorch 1.1.0, and conduct the experiments on an Enterprise Linux Server with 40 Intel(R) Xeon(R) CPU cores (E5-2640 v4 @ 2.40 GHz and 252G memory), and a single NVIDIA Tesla with 24 GB memory size.

A.1.2 Pre-training Module

The dimension of the BERT output embedding is 768, which is then fed into a one-layer MLP in Eq.(2) to get the corresponding outputs:

$$\text{MLP}(\mathbf{X}) = \text{Tanh}(\mathbf{W}^T\mathbf{X}), \tag{15}$$

where $\mathbf{W} \in \mathbb{R}^{768\times512}$.

The two-layers MLP on top of the metric function in Eq.(9) is defined as follows:

$$\text{MLP}(\mathbf{X}) = \tanh(\mathbf{W}_2^T\text{LeakyReLU}(\mathbf{W}_1^T\mathbf{X})), \tag{16}$$

where $\mathbf{W}_1 \in \mathbb{R}^{21\times21}$ and $\mathbf{W}_2 \in \mathbb{R}^{21\times1}$.

In Eq.(6), we use 21 kernels, where μ is from 0 to 1 with interval 0.05. The kernel with $\mu = 0.0$ and $\sigma = 10^{-3}$ corresponds to the exact matching kernel, while σ is set as 0.1 for other kernels capture the semantic matches at different scales.

The margin m of the triplet loss in Eq.(10) is set to 1.0.

TABLE 6
Features extracted for GBDT model. p: paper, a: an author in p, c: a candidate expert whose name is similar as a.

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

A.1.3 Adversarial Fine-tuning Module

The two-layers MLP for the domain discriminator in Eq.(12) is defined as:

$$\text{MLP}(\mathbf{X}) = \mathbf{W}_2^T\text{LeakyReLU}(\mathbf{W}_1^T\mathbf{X}), \tag{17}$$

where $\mathbf{W}_1^T \in \mathbb{R}^{512\times100}$ and $\mathbf{W}_2^T \in \mathbb{R}^{100\times2}$.

A.2 Training and Test Settings

A.2.1 Evaluation of the Pre-training Module.

Pre-training is performed only on the AMiner dataset.

Training Settings. For the AMiner dataset shown in Table 1, we first filter the experts with less than 6 papers to satisfy the best number of the sampled papers for each expert. For each expert in the remaining AMiner dataset, we randomly sample $L = 6$ papers to comprise an expert instance. For each anchor expert instance, we sample 1 positive counterpart together with 9 negative counterparts, which results in 4,800 positive instance pairs and 43,200 negative instance pairs in total. For embedding a paper, we use [CLS] + title + keywords + name + organization + venue + [SEP] as the BERT input. The maximal length of the input tokens is set as 208.

Test Settings. Author identification and paper clustering are two intrinsic tasks for testing the pre-training module.

- **Author Identification:** follows the second task of the name disambiguation competition[10]. Thus, we adopt the same test set as the competition. Since the average number of the experts per name in the dataset is about

$1,806/100 \approx 18$, we randomly sample 17 negative experts together the ground truth expert to comprise the candidate list for a queried paper. The maximal number of papers sampled for each candidate expert is set to 100. For testing, we first leverage the pre-trained expert encoder g to generate the paper embedding for each candidate expert, then use the pre-trained metric function f to measure the similarity between a paper and each candidate expert. Finally, we rank all the candidate experts for a paper according to their similarity scores and return the top expert as the expert to be linked.

- **Paper Clustering:** follows the first task of the name disambiguation competition[11]. We directly use its test set. For testing, we apply the pre-trained expert encoder g to get the paper embeddings and then use the HAC algorithm to cluster the papers belonging to a same expert together. For a fair comparison, the true number of clusters in each name are provided to the HAC algorithm.

Implementation of baselines. The heuristic features of GBDT [11], [23] are specified in Table 6. Apart from GBDT, We implement other baselines following their released code and settings. The training and test settings of all the baselines are the same as COAD (pre-train).

- **Camel** [47]: https://github.com/chuxuzhang/code_Camel_WWW2018
- **HetNetE** [5]: https://github.com/chentingpc/GuidedHeteEmbedding
- **CONNA** [3]: https://github.com/BoChen-Daniel/TKDE-2019-CONNA
- **louppe at el** [26]: https://github.com/glouppe/paper-author-disambiguation
- **Zhang et al** [46]: https://github.com/baichuan/disambiguation_embedding
- **G/L-Emb** [50]: https://github.com/neozhangthe1/disambiguation/

Both Camel and HetNetE define an additional loss function on the indirect relationships between a paper and an expert generated by the pre-defined meta-paths. We ignore this part for the following reasons. Since author identification in their work aim to predict the authors of an anonymous paper, name ambiguity is not the key problem to be tackled. Thus they can collect all the papers published in some related venues as the training data. The connectivity of the resultant heterogeneous graph is good enough to find the indirect relationships between any two nodes. However, to disambiguate experts with same names, we only collect the papers with the same author names. The resultant graph is too sparse to find the indirect relationships.

A.2.2 Evaluation of the Fine-tuning Module

We evaluate of the fine-tuning module on two external datasets, News and LinkedIn, by the extrinsic task of external expert linking. We first fine-tune COAD (pre-train) on both AMiner and the unlabeled external dataset, and evaluate it on the manually-labeled test set.

News. The fine-tuning settings for news are as follows:

- **Training.** We use 20,658 news articles for fine-tuning. We divide the contextual text of a name in a news article into sentences and treat each sentence as a piece of support information. The maximal length of the input tokens for both the shared encoder and the private encoder is set as 64.
- **Test.** We annotate 1,622 names in news articles with linkages to AMiner experts for testing. We choose the candidates by name variants and sample up to 100 papers for each candidate.

LinkedIn. The settings for LinkedIn are as follows:

- **Training.** We use 50,000 LinkedIn homepages for fine-tuning. We select three common semi-structured attributes including affiliation, skills and summary in a homepage as the support information. The affiliation and the concatenated keywords in skills are separate pieces of support information. The long-text summary is divided into multiple pieces of support information. The maximal length of the input tokens for both the shared encoder and the private encoder is set as 64.
- **Test.** We annotate 1,329 linkages between LinkedIn users and AMiner experts for testing. Other settings are the same as News.