OAG: Toward Linking Large-scale Heterogeneous Entity Graphs

Fanjin Zhang[†], Xiao Liu[†], Jie Tang[†], Yuxiao Dong[‡], Peiran Yao[†], Jie Zhang[†], Xiaotao Gu[†],

Yan Wang[†], Bin Shao[‡], Rui Li[‡], and Kuansan Wang[‡]

[†]Tsinghua University [‡]Microsoft Research
[†]Beijing National Research Center for Information Science and Technology, China
{zfj17,liuxiao17,ypr15,j-z16,yan-w16}@mails.tsinghua.edu.cn,jietang@tsinghua.edu.cn
{yuxdong,binshao,Kuansan.Wang}@microsoft.com,xiaotao2@illinois.edu,lerain@gmail.com

ABSTRACT

Linking entities from different sources is a fundamental task in building open knowledge graphs. Despite much research conducted in related fields, the challenges of linking *large-scale heterogeneous entity graphs* are far from resolved. Employing two billion-scale academic entity graphs (Microsoft Academic Graph and AMiner) as sources for our study, we propose a unified framework — LinKG — to address the problem of building a large-scale linked entity graph. LinKG is coupled with three linking modules, each of which addresses one category of entities. To link word-sequence-based entities (e.g., venues), we present a long short-term memory network-based method for capturing the dependencies. To link large-scale entities (e.g., papers), we leverage locality-sensitive hashing and convolutional neural networks for scalable and precise linking. To link entities with ambiguity (e.g., authors), we propose heterogeneous graph attention networks to model different types of entities. Our extensive experiments and systematical analysis demonstrate that LinKG can achieve linking accuracy with an F1-score of 0.9510, significantly outperforming the state-of-the-art. LinKG has been deployed to Microsoft Academic Search and AMiner to integrate the two large graphs. We have published the linked results—the Open Academic Graph (OAG)[1], making it the largest publicly available heterogeneous academic graph to date.

CCS CONCEPTS

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

KEYWORDS

Entity Linking, Name Ambiguity, Heterogeneous Networks, OAG

*Xiaotao and Rui are now at UIUC and Google. This work was done when they were at Tsinghua and Microsoft.

[1]https://www.openacademic.ai/oag/

KDD '19, August 4–8, 2019, Anchorage, AK, USA

ACM ISBN 978-1-4503-6201-6/19/08…$15.00
https://doi.org/10.1145/3292500.3330785

ACM Reference Format:
Fanjin Zhang, Xiao Liu, Jie Tang, Yuxiao Dong, Peiran Yao, Jie Zhang, Xiaotao Gu, Yan Wang, Bin Shao, Rui Li, and Kuansan Wang. 2019. OAG: Toward Linking Large-scale Heterogeneous Entity Graphs. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3292500.3330785

1 INTRODUCTION

Entity linking, also called ontology alignment and disambiguation [12], is the task of determining the identity of entities across different sources. The problem of *entity linking* is related to entity matching [20, 25, 36], entity resolution [2, 3], web appearance disambiguation [1, 11], name identification [16], object distinction [37], and name disambiguation [8, 30, 40], and has been extensively studied for decades by different communities.

However, despite the bunch of research, the challenges of linking Web-scale heterogeneous entity graphs from different sources are far from resolved. First, the Web-based entity graphs are usually heterogeneous, in the sense that they consist of various types of entities, such as author, paper, and venue entities in academic graphs. Second, it is very common to observe ambiguous entity mentions in entity graphs. For example, there are more than 10,000 authors with the name "James Smith" in Microsoft Academic Graph (MAG) [26]. Finally, the scale of entity graphs on the Web is usually large, with billions of entities.

In this work, we employ two academic entity graphs (MAG and AMiner [33]) to conduct a systematical analysis for the problem of linking large-scale heterogeneous entity graphs. A straightforward method to address this problem is to find the entity alignments using heuristic rules [15]. However, such an ad-hoc strategy cannot be flexibly generalized to other scenarios. Several efforts have also been made to find the alignments via learning algorithms such as neural networks [17] and probabilistic frameworks [24]. However, the complexity of these methods (usually $O(n^2)$) is very expensive, making them not scalable for handling large graphs in practice. Additionally, recent works attempt to combine human annotators into the loop of entity linking [40, 41]. Still, these methods are insufficient to handle large heterogeneous graphs.

Figure 1 illustrates an example of linking the heterogeneous MAG and AMiner academic graphs. As can be seen, there are 16,392 authors with the name "Jing Zhang" in AMiner and 7,170 in MAG. The entity linking methods must address this inherent ambiguity. In addition, both graphs consist of different types of entities, such

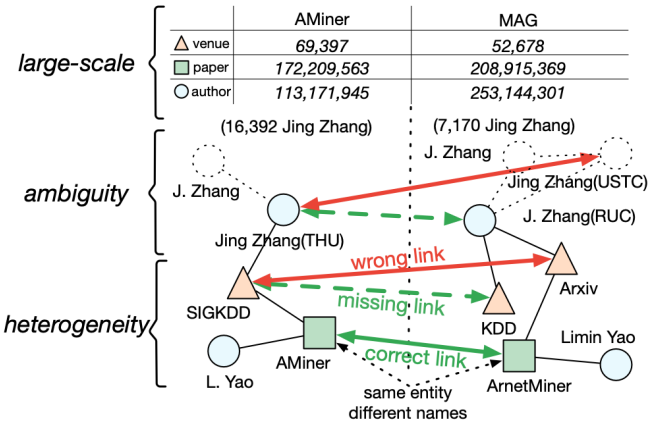

Figure 1: An illustrative example of the large-scale heterogeneous entity linking problem. The table above displays the overall 0.7 billion entities in this *large-scale* graph. The graph in the middle illustrates *ambiguity* in author linking, where name distinction and affiliation shift exist. At the bottom, node and attribute *heterogeneity* increase great difficulty in linking.

as authors, venues, and papers. How to leverage the heterogeneous structure to improve the linking accuracy is largely unexplored. Last, the AMiner graph comprises more than 285,450,905 entities, and MAG covers over 462,112,348 entities. Therefore it is desired to design efficient and scalable techniques to conquer the computational challenge.

We propose a unified framework—LinKG—to address the above challenges, toward building a large-scale linked entity graph. The framework is coupled with three linking modules, each of which corresponds to one type of entity: venues, papers, and authors. To link venues, which are coarse-grained and word-sequence dependent entities, we customize the long short-term memory networks [10] (LSTM) to capture the sequential dependency in venue names; To link paper entities, which are relatively less ambiguous but at a very large scale, we present a locality-sensitive hashing [6] and convolutional neural network [14] (CNN) based techniques for fast and accurate matching; To link large-scale author entities, which are highly ambiguous, we propose heterogeneous graph attention networks (HGAT). In addition to leveraging information from each graph, we also incorporate the linked venue and paper entities in previous steps into HGAT.

To summarize, our work makes the following contributions.

- First, we propose to study the entity linking problem in two large-scale heterogeneous entity graphs, in which each type of entity has different properties and thus the linking of them faces different challenges. We develop an effective and efficient framework—LinKG, which leverages state-of-the-art deep neural networks for linking heterogeneous entities.
- Second, we conduct large-scale linking experiments between the MAG and AMiner graphs. The results show that our framework can achieve very high accuracy of 0.9926 for linking venue entities, 0.991 for papers, and 0.9741 for authors. We also conduct extensive experiments to demonstrate our design choice of each module in the framework.
- Finally, with the linking results, we published the Open Academic Graph (OAG). OAG consists of 0.7 billion entities and 2 billion relationships, making it the largest public academic data to date[2]. The dataset can be used for various research topics, such as network science and graph mining (collaboration and citation), text mining and natural language processing (title and abstract), science of science, computational social science, etc.

2 RELATED WORK

In this section, we review relevant literature about entity linking. Entity linking, also known as data integration, record linkage etc., is a classical problem which was put forward more than six decades ago [7]. Some literature reviews can be found in [7, 25].

There are various approaches to tackle this problem. Li et al. [15] argue for rule-based methods and develop a rule discovery algorithm. Another important thread of research is based on machine learning algorithms. Tang et al. [31] regard entity matching as minimizing Bayesian risk of decision making. Some works [22, 23] attempt to use less labeled data and employ semi-supervised or unsupervised matching algorithms. For example, Rong et al. [22] transfer the entity matching problem to a binary classification problem and use pairwise similarity vectors as training data. Wang et al. [35] present a factor graph model to learn the alignment across knowledge bases. For data integration across social networks or other networks, some [29, 39] incorporate network structure to develop effective algorithms. Zhang et al. [39] propose COSNET, an energy-based model which considers global and local consistency of multiple networks.

Recently, network embedding based approaches [9, 17, 18] have been proposed to address the user alignment problem across social networks. For example, Liu et al. [17] propose an optimization framework to learn user embedding simultaneously across networks, by considering hard and soft constraints. Heimann et al. [9] propose an embedding-based solution for graph alignment based on the factorization of a user-similarity matrix of two graphs. However, for large-scale graphs, constructing the similarity matrix is very expensive. Similarly, Zhang et al. [38] propose MEgo2Vec, which employs attention mechanisms and graph neural networks elaborately to match ego networks of two candidate users.

However, few studies aim to find a unified solution for large-scale heterogeneous entity linking. For such a problem, one must consider efficiency, heterogeneity, ambiguity issues. In this paper, we propose a unified framework to deal with these issues. We present a hashing-based method to efficiently find linkings with less ambiguity. We use LSTM to perform fuzzy-sequence linking and CNN to perform fine-grained text matching . Finally, based on results obtained by the aforementioned approaches, we use graph attention networks to perform linking with more ambiguity.

3 PROBLEM DEFINITION

In this section, we formalize the problem of entity linking across heterogeneous entity graphs.

[2]The dataset has been downloaded more than 40,000 times over the past months (until 2019-02-03).

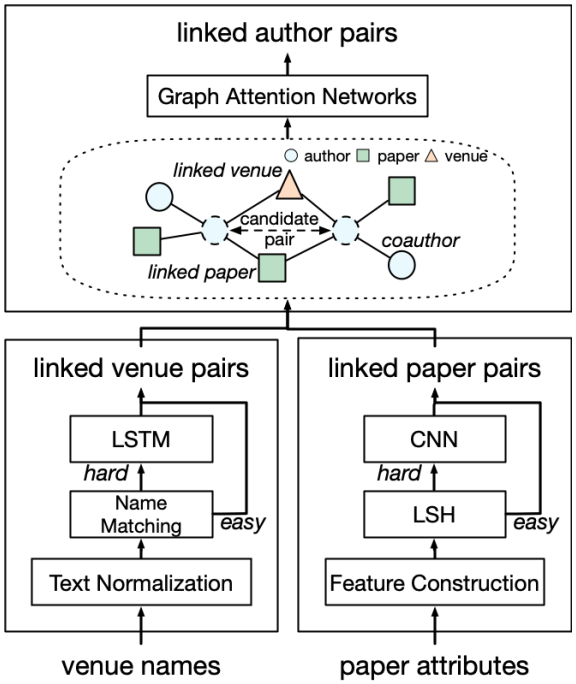

Figure 2: The architecture of LinKG.

Definition 3.1. Heterogeneous Entity Graph (HEG) (also known as the heterogeneous information network (HIN) [28]). A HEG is defined as a graph $HG = \{E, R\}$ where each entity $e \in E$ and each relation $r \in R$ are associated with type mapping functions $\tau(e) : E \rightarrow C$ and $\phi(r) : R \rightarrow D$, respectively. C and D represent the entity and relation types with $|C| > 1$ and $|D| > 1$.

For example, an academic graph is a heterogeneous entity graph comprises multiple types of entities: authors, papers, and venues. Its relation set D includes 1) the *authorship* relation between authors and papers, 2) the *paper-publish-in-venue* relation between papers and venues, 3) the *co-authorship* relation between authors, and 4) the *author-publish-in-venue* relation between authors and venues.

Problem 3.1. Entity Linking across HEGs. *Given two heterogeneous entity graphs HG_1 and HG_2, the goal is to generate entity linkings $L = \{(e_1, e_2)|e_1 \in HG_1, e_2 \in HG_2\}$ such that e_1 and e_2 represent exactly the same entity in HG_1 and HG_2.*

In this work, we focus on the problem of entity linking across two heterogeneous academic entity graphs: Microsoft Academic Graph and AMiner, each of which consists of author, paper, and venue entities. The goal is to link these three types of entities for the two large-scale graphs. The proposed approaches are general and can be extended to handle other heterogeneous networks.

4 THE *LINKG* FRAMEWORK

We present a unified framework, *LinKG*, which is coupled with three modules for linking venue, paper, and author entities, respectively, across the MAG and AMiner heterogeneous graphs. In each module, we present techniques that are capable of addressing the unique challenges in linking each type of entities. Figure 2 illustrates the overview of the LinKG framework.

(1) *Venue linking*. The venue entity linking task is formulated as: given the full names of venues in each graph, the goal is to connect the same venues from both graphs. Though additional information can be utilized for venue linking, we find that the single usage of venue full names is simple, efficient, and effective. The venue linking module consists of two technical components: venue name matching and sequence encoding empowered by long short-term memory networks.
(2) *Paper linking*. To link paper entities, we fully leverage the heterogeneous information, including a paper's title and publication year, as well as the other two types of entities, i.e., its authors and publication venue. Notice that both graphs contain hundreds of millions of paper entities and billions of relations between papers and their authors. Thus, we propose to leverage 1) the hashing technique (e.g., locality-sensitive hashing) for fast processing and 2) the convolution neural networks for effective linking.
(3) *Author linking*. To link author entities, we generate a heterogeneous subgraph for each author. One author's subgraph is composed of his or her coauthors, papers, and publication venues. Moreover, we incorporate the venue and paper linking results from the first two modules into author linking. This linking task faces not only the scalability issue but also the long-standing challenge of "name disambiguation" [8, 30]. To tackle them, we present a heterogeneous graph attention network based technique for linking author entities.

Due to the different properties of the three types of entities, we design three different neural networks for addressing their associated challenges. Empirical results in Section 5 demonstrate the effectiveness of our design and modeling choices over other alternative methods.

4.1 Linking Venues — Sequence-based Entities

The first category entity we are going to deal with is word-sequence dependent entity — i.e., venues in the academic graph. Ideally, all attributes associated with venues can be leveraged for linking them across the two graphs, including the venue full name, keywords, and its publications, as well as those publications' authors. However, many journals in the dataset publish millions of papers that are also associated with millions of authors, making it difficult to utilize them straightforwardly and efficiently.

Furthermore, the keyword attribute also contributes little to the distinction between venues in similar fields. For instance, the two journals *'Diagnostic and interventional imaging'* and *'Journal of Diagnostic Imaging and Interventional Radiology'* share exactly the same keywords "diagnostic", "interventional", and "imaging", resulting in a very high similarity by using common similarity metrics, such as Jaccard Index.

Name Matching. More importantly, it turns out that the direct matching between two graphs' venue entities by using their full names and abbreviations can link more than 27,000 venue pairs, corresponding to the majority of the final linking results. The challenging part of this task lies in the remaining venues that cannot be matched directly. In specific, these venues usually have the following characteristics:

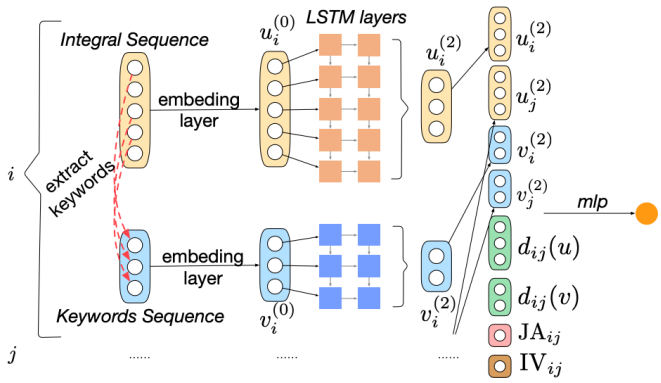

Figure 3: The LSTM model for modeling sequencial dependency in venue full names.

(1) Word order inversion: The same journal is observed with different word orders in two graphs, such as *Journal of Shenzhen University* and *Shenzhen University Journal*.
(2) Extra or missing prefix or suffix: For many venues, their names contain additional annotations, such as *Proceedings of the Second international conference on Advances in social network mining and analysis*.

LSTM. Name matching is not able to handle those venues that cannot be exactly matched. We observe that the relative word or keyword sequences in the full name is generally preserved. Therefore, we propose to model both the *Integral Sequence* and *Keyword Sequence* in venue names. Integral Sequence is the raw word sequence from venue names, and Keyword Sequence is derived from the keywords extracted from the Integral Sequence. We present an enhanced long short-term memory networks (LSTM) to address the issues above.

Given a venue i in one graph, let $u_i^{(0)}$ be the input Integral Sequence and $v_i^{(0)}$ be the Keyword Sequence, we can have:

$$u_i^{(2)} = \text{lstm}_{\text{u2}}(\text{lstm}_{\text{u1}}(u_i^{(0)})),\ v_i^{(2)} = \text{lstm}_{\text{v2}}(\text{lstm}_{\text{v1}}(v_i^{(0)})) \quad (1)$$

where *lstm* denotes the LSTM layers. Then we calculate the differences d_{ij} with venue j from the other graph:

$$d_{ij}(u) = u_i^{(2)} - u_j^{(2)},\ d_{ij}(v) = v_i^{(2)} - v_j^{(2)} \quad (2)$$

Finally, we concatenate all of them with Jaccard Index (JA_{ij}) and the number of inversion pairs in Keyword Sequence (IV_{ij}), and employ fully-connected layers to calculate the similarity between the two venues:

$$\begin{aligned} s_{ij} &= [u_i^{(2)}, v_i^{(2)}, u_j^{(2)}, v_j^{(2)}, d_{ij}(u), d_{ij}(v), \text{JA}_{ij}, \text{IV}_{ij}] \\ y_{ij} &= fc(s_{ij}) \end{aligned} \quad (3)$$

where fc denotes the fully-connected layers and y is the similarity between venue i and j. In training, we use venues from labeled candidate pairs to learn parameters in the LSTM layers and fc layer; and in matching, we use the learned parameters to predict the similarity between two venues.

4.2 Linking Papers — Large-scale Entities

The second category entity is of large-scale. This makes the paper linking problem faces several challenges. First, the overall volume of academic publications at each of the graphs is at the scale of hundreds of millions, requiring efficient linking techniques. Second, it is commonly observed from not only the MAG and AMiner graphs but also the official publishers (such as ACM and IEEE) that paper titles are often truncated if they contain punctuation marks, such as ':' and '?', making the linking task more challenging. Finally, there also exist papers with exactly the same titles even in the same venue in both graphs, such as the two different "robust influence maximization" papers published in KDD 2016, making this task harder and harder.

To address these challenges, we propose to leverage the hashing technique and convolution neural networks for linking paper entities across the two graphs.

Locality-sensitive Hashing. If we compare all possible pairs of papers from both sides, the matching complexity is at least $O(n^2)$, where n is the number of papers in each graph. To reduce the computational cost, we propose to leverage the hashing technique to match paper entities. Specifically, we use the locality-sensitive hashing (LSH) algorithm, which is widely used for efficient nearest neighbor search.

An intuitive way to map titles to binary codes is to use word one-hot encoding. However, this will make binary representations high-dimensional, and mapping them to low-dimensional binary space using LSH is likely to lose information. We adopt Doc2Vec [13] to first transform titles to low-dimensional real-valued vectors. Then LSH is applied to further map real-valued vectors to binary codes.

For efficient linking, we directly query the indexed binary codes and obtain uniquely matched papers in $O(1)$ time. We find that recall will get much lower when leveraging other heterogeneous attributes, such as authors, venues. Therefore, we just employ title hashing to deal with easy cases for paper linking.

Convolution Neural Networks (CNN). LSH can match papers from the large-scale MAG and AMiner graphs efficiently. However, due to its probabilistic nature, the information loss in LSH, e.g., during its randomized projection stage, would result in missing paper candidates. Therefore, we also propose to leverage a more elaborate method, that is, convolution neural networks, to capture fine-grained matching signals for those unlinked papers.

The CNN based linking strategy consists of three steps: 1) candidate paper pair search; 2) paper similarity matrix construction; 3) CNN-based pairwise similarity learning.

To avoid n^2 pairs of candidate papers, we employ the inverted index technique based on title keywords to filter candidate paper pairs. We then construct two similarity matrices for each candidate pair, by using their paper titles and authors, respectively, as the input of the CNN model. Each element $z_{ij}^{(0)}$ in the similarity matrix is set to 1 if the i-th word (name initial) and j-th word (name initial) in two paper's titles (authors) are the same, -1 otherwise. Instead of concatenating the two matrices, we model them individually in the first CNN layers. This is because there is no need to measure cross-entity similarities, such as between one paper's title and the

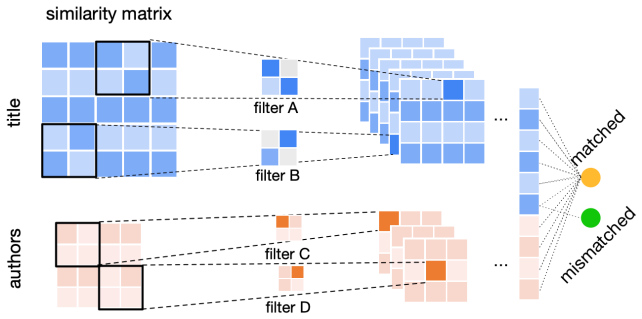

Figure 4: The CNN model for modeling heterogeneous attributes of paper entities.

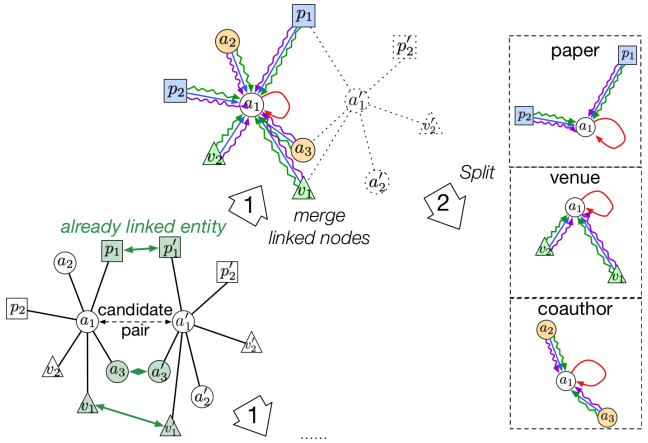

Figure 5: The heterogeneous graph attention networks for modeling heterogeneous structures around authors. p_1 and $p_1^{'}$, v_1 and $v_1^{'}$ are conflated as the same entity in previous steps, respectively. The boxes in the right represent the different roles played by heterogeneous entities in author linking, which are captured by different attention mechanisms.

other one's authors. The overall CNN architecture for paper linking is shown in Figure 4.

For the first layer of CNN, the n-th filter $w^{(1,n)}$ scans over each input similarity matrix $z^{(0)}$ to generate a feature map $z^{(1,n)}$:

$$z_{x,y}^{(1,n)} = \sigma\Big(\sum_{i=0}^{r_n-1}\sum_{j=0}^{r_n-1} w_{i,j}^{(1,n)} \cdot z_{x+i,y+j}^{(0)} + \theta^{(1,n)}\Big), \tag{4}$$

where r_n and $\theta^{(1,n)}$ denote the size and the bias of the n-th filter, respectively. Herein, the square filter and ReLU [5] are adopted. Moreover, multiple filters are used to capture different similarity patterns to deal with the data heterogeneity issue.

The following layers are convolutional or pooling layers for capturing the higher-order matching features. After flattening the hidden layer matrix to a dense vector and concatenating the two vectors learned from title and author similarity matrices, Multi-Layer Perception (MLP) is used to produce the final matching score. Finally, softmax function is utilized to output the matching probability and cross entropy is used as the objective.

In our framework, we first leverage locality-sensitive hashing to link easily matched paper pairs. For those papers which can not obtain confident matching results via LSH, we further employ CNN to perform fine-grained linking.

4.3 Linking Authors — Ambiguous Entities

In this subsection, we introduce the author linking module, which is more challenging than linking papers and venues, due to the author name ambiguity issue. The good news is that linked entities (i.e., venues and papers) in previous steps can help alleviate this problem. Similar to the paper linking task, it is impractical to link hundreds of millions of authors from two graphs. Therefore, the first step is to generate candidate pairs for authors if they have similar names; Second, for each author in the candidate pair, we construct a heterogeneous ego subgraph, and two ego subgraphs can be connected with each other if they share common venues or papers that have been linked before; Finally, a heterogeneous graph attention network is applied for determining author matching.

Paired Subgraph Construction. For each author in a candidate pair, its direct (heterogeneous) neighbors are selected, including its coauthors, papers, and venues. The left part of Figure 5 illustrates an example of the construction process. If two authors' papers or venues have been linked in previous steps, we can connect the two authors' subgraphs together. Note that we construct a fixed-size paired subgraph based on collaboration and publication frequencies in order to better feed into graph neural networks. In addition, we also include coauthors' papers and venues for constructing the paired subgraph (i.e. a two-hop ego network).

Heterogeneous Graph Attention Networks (HGAT). The common neighbors of candidate author pairs (such as papers and venues) could provide evidence for author pair matching. We propose using graph attention networks to combine all the information by aggregating needed pieces of information from neighbors. Next we will introduce pre-training, encoder layers and output layers of this model.

We pre-train input features for each entity based on both semantic and structure information. For semantic features, we first train a skip-gram word embedding model [19] for all words on the AMiner publication corpus (including titles, authors, abstracts, etc.) . Then the semantic embedding of each entity is obtained by averaging over its associated words' embeddings. For structure features, we train the LINE model [32], due to its simplicity, on a large heterogeneous entity graph (HEG) and get the structural embeddings for each entity. Two types of embeddings are concatenated together as entity input features h to graph attention networks.

Encoder layers. The encoder is actually multiple graph attention layers [34]. The goal of graph attention is to learn the attention coefficient $\text{attn}(e_i, e_j)$, which implies the aggregation weight of source entity e_j's embedding on target entity e_i. The attention coefficient is learned by self-attention mechanism, i.e.,

$$o_{ij} = \text{attn}(Wh_i, Wh_j) \tag{5}$$

where o_{ij} indicates the importance of node e_j's features to node e_i, h_i is node e_i's input features, and W is a shared projection matrix. By utilizing the subgraph structure, the graph attention layer only needs to compute o_{ij} for nodes $e_j \in \mathcal{N}_i$, where $\mathcal{N}_i$ is

the neighborhood of node e_i. Then o_{ij} can be normalized across all possible e_j by using softmax function. In this work, the normalized attention coefficient α_{ij} is specified as below:

$$\alpha_{ij} = \frac{\exp(\text{LeakyReLU}(c_{\tau(e_i)}^{\top} W h_i + c_{\tau(e_j)}^{\top} W h_j))}{\sum_{k \in \mathcal{N}_i} \exp(\text{LeakyReLU}(c_{\tau(e_i)}^{\top} W h_i + c_{\tau(e_k)}^{\top} W h_k))} \quad (6)$$

where $c_{\tau(e_i)}$ denotes the attention parameter vector of entity e_i's type $\tau(e_i)$. Note that different from the original GAT [34], we use different attention parameters for each type of entities because different types of entities play different roles in author linking. The heterogeneous attention mechanism are plotted in Figure 5. Then we employ the multi-head attention to generate node e_i's output embedding $h_i^{'}$ as

$$h_i^{'} = \Big\Vert_{k=1}^{K} \sigma\Big(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^k W^k h_j\Big) \quad (7)$$

where $\Vert$ represents concatenation, σ is the activation function and K is the head number of one layer. In our model, we use two graph attention layers to instantiate the encoder part.

Output layers. Passed through the graph encoder, each node has a hidden embedding by aggregating its near neighbors. Let $\hat{h}_{MAG}$ and $\hat{h}_{AMiner}$ denote the embeddings of two focal authors in a candidate pair. We fuse these two embeddings into one vector and then use two fully-connected layers to produce output representation for each pair,

$$y = fc(\hat{h}_{MAG} \otimes \hat{h}_{AMiner}) \quad (8)$$

where $fc(\cdot)$ denotes the fully-connected layers and we choose the element-wise multiplication operator as the vector operation $\otimes$. Finally, we optimize the negative log-likelihood loss function by comparing the output vectors with ground truths.

4.4 Further Discussions

In this section, we use academic entity graphs as the example to explain how the proposed framework LinKG deals with the three challenges: large-scale, heterogeneity, and ambiguity, in our problem. The proposed methods are general and can be also applied to various different heterogeneous entity graphs, for example the heterogeneous social networks [39]. The method for linking venues can be used to link entities with word-sequence information, while the LSH method for linking papers can be used to link entities of large scale. In case *recall* is also very important, the combination of LSH and CNN could achieve a better trade-off between efficiency and effectiveness. The method for linking authors leverages the graph structure information and also linking results of the other entities, thus could achieve a high linking accuracy.

5 EXPERIMENT

We evaluate our solution in the context of two heterogeneous entity graphs: MAG and AMiner. Microsoft Academic Graph (MAG) consists of 208,915,369 papers, 52,678 venues, 253,144,301 authors and so on, which is a snapshot taken in Nov. 2018. AMiner consists of 172,209,563 papers, 69,397 venues, 113,171,945 authors and so on, which are snapshots taken in July 2018 (for venues and authors) or Jan. 2019 (for papers). Next, we present our main experimental results on venue linking, paper linking and author linking. The OAG data set is publicly available.[1] The codes and training data are available at https://github.com/zfjsail/OAG.

5.1 Experiment Setup

Datasets We construct the following datasets to evaluate venue linking, paper linking and author linking.

- **Venue datasets**: We generate a relatively hard dataset, by excluding trivial cases that can be matched by full name or short name directly. We use Jaccard Index to calculate similarity between venues and manually label 1,202 pairs for training and test, 30% of which are used for test set.
- **Paper datasets**: Like venue datasets above, we construct a difficult paper linking dataset. We create two paper sets: clean paper set CP and noisy paper set NP. The goal is to link papers between CP and NP. The clean set contains 46,170 papers extracted from AMiner. The NP dataset is constructed by adding noises to each paper. Thus, each original paper and its corresponding noisy paper become a positive matching pair. The method to add noise is based on differences in existing matched papers, such as different author name formats and missing title words. The negative paper pairs we generate share some common keywords to increase linking difficulty. We use 20% noisy papers for test (to find corresponding matched clean papers).
- **Author datasets**: We use the following two ways to generate positive author pairs. (1) sample author pairs with unique names and matched papers. (2) sample author pairs both of which are top coauthors (w.r.t. collaboration frequency) of strictly matched author pairs by rules. We use negative pairs which cannot be matched by strict rules but one of which can be matched to some other author by rules. Negative pairs should also have similar names. We construct 10,000 positive pairs and 10,000 negative pairs respectively. The training/validation/test split is 2:1:1. To increase the task difficulty, we also add some noises to the dataset. For example, we mask some authors' affiliations or replace a fraction of authors' papers/venues with random papers/venues.

Comparison Methods We compare our proposed methods with the following methods. Some methods with the same name are designed differently according to different entity characteristics.

- **Keyword**:
  - For venues: We use TF-IDF weighted Jaccard index to measure the similarity of two venue names.
  - For papers: Given a paper $p \in NP$, we use its title keywords to find candidate papers in CP by inverted index table and then re-rank these candidates by edit distance between two titles and character-level similarity of its author names.
  - For authors: We match two authors if and only if they have exact the same full name.
- **SVM**:
  - For venues: We use similarity scores of venue integral sequences and keyword sequences as input features.
  - For papers: We use similarity scores of paper titles and authors as input features.
  - For authors: We use similarity scores of author names, affiliations, venues, papers and coauthors as input features.

  For detailed feature definition, please refer to Appendix. A.2.2.

Table 1: Results of linking heterogeneous entity graphs. "–" indicates the method does not support the entity linking.

Methods		Keyword	SVM	Dedupe	COSNET	MEgo2Vec	LinKG$_C$	LinKG$_L$	LinKG
Venue	Prec.	80.15	81.69	84.25	–	–	84.67	**91.16**	**91.16**
	Rec.	83.76	83.45	80.92			85.81	**87.58**	**87.58**
	F1	81.91	82.56	82.55			85.23	**89.33**	**89.33**
Paper	Prec.	**99.57**	87.73	99.30	–	–	96.60	89.85	96.70
	Rec.	25.31	85.42	87.09			94.55	89.71	**94.70**
	F1	40.36	86.56	92.80			95.57	89.78	**95.69**
Author	Prec.	44.48	84.70	50.65	91.73	91.03	81.30	84.92	**95.37**
	Rec.	80.63	92.22	85.46	85.33	90.82	84.95	**94.75**	93.48
	F1	57.33	88.30	63.60	88.42	90.92	83.09	89.57	**94.42**
Overall	Prec.	80.22	86.54	82.26	91.73	91.03	91.06	88.19	**96.11**
	Rec.	45.71	87.70	86.38	85.33	90.82	91.05	91.38	**94.11**
	F1	58.23	87.12	84.27	88.42	90.92	91.05	89.76	**95.10**

- **Dedupe [4]**[3]: This is an open-source toolkit to perform deduplication, record linkage on structured data. Basically, it uses blocking [27] technique to reduce the number of record comparisons and employs active learning and logistic regression to learn weights of attribute features.
- **COSNET [39]**: This method is a factor graph model which considers user pairwise features as local factors and the relationship between user pairs as correlation factors. An efficient dual decomposition method is developed to optimize original objective. This method is used for author linking problem.
- **MEgo2Vec [38]**: This method feeds ego networks of two users as inputs, and uses graph neural networks to map the subgraphs to vectors and then predict the labels of user pairs. This method is used for author linking problem.
- **LinKG$_C$**: For each candidate pair, this method first constructs similarity matrices and then use CNN to map these matrices to similarity vectors. Finally, similarity vectors of different attributes are concatenated and passed into fully-connected layers to output matching scores.
- **LinKG$_L$**: In this method, each attribute is treated as a word sequence. The content of each attribute is embedded as vector by LSTM. For each pair, vectors of the same attribute are merged (by concatenation, subtract, etc.) into similarity vectors, which are fed into fully-connected layers to predict labels.
- **LinKG**: Our best model is indicated by LinKG, which uses different methods tailored for different entity linking problems.
  - For venues, LinKG$_L$ is our best model. We use LSTM to model venue features as described in Sec. 4.1.
  - For papers, we first leverage LSH to perform fast paper linking, and then use CNN to cope with harder cases which can not matched via LSH (Cf. 4.2).
  - For authors, the best model is described in Sec. 4.3. We employ heterogeneous graph attention networks on constructed paired subgraphs to generate prediction results. We also introduce some discriminative features (such as paper/venue matching ratio of two authors) to enhance results.

[3]https://github.com/dedupeio/dedupe

5.2 Overall Results

Table 1 shows the overall linking performance of different methods. Our method (LinKG) consistently outperforms other alternatives (+4.05%-36.87% in terms of F1-score). The overall F1-score is weighted by the number of test samples on different linking problems (i.e. 361, 9234 and 5000 test pairs for venues, papers, authors respectively). Next we compare and analyze results on the linking of venues, papers and authors one by one.

For venue linking, LinKG (namely LinKG) outperforms other methods. Keyword and SVM perform poorly owing to problems mentioned above in Section 4.1, such as word order reversion or mismatched prefix and suffix. LinKG$_C$ can also achieve good performance for venue linking, because CNN is capable of capturing word order matching pattern. The advantage of LSTM is that it can capture not only word order information but also coarse-grained information. Besides, compared with CNN, LSTM can process variable-sized sequences while CNN cannot.

For paper linking, LinKG and LinKG$_C$ obviously outperform other methods. Dedupe can achieve high precision but has a low recall, which indicates it prefers a high threshold for the classifier. Compared with Keyword and SVM, CNN uses low-level word similarity matrix and can learn fine-grained matching patterns automatically.

For author linking, our method LinKG, i.e. employing graph attention networks by combining linked venue and paper results, achieves the best performance among other methods. By full name matching, Keyword performs poorly because different authors with the same name are linked incorrectly and the same author is likely to have different name formats. LinKG$_C$ and LinKG$_L$ take many author attributes, including affiliations, papers, and venues, but perform worse than methods incorporating graph structures, such as COSNET and MEgo2Vec. For COSNET, prediction of an author pair will be influenced by the prediction of their neighbor pairs thus it may suffer from the error propagation problem. Compared with MEgo2Vec, our method uses simpler attention mechanism and is able to distinguish the effects of different types of entities. Furthermore, our method can incorporate the effect of distant neighbors besides direct neighbors while MEgo2Vec can not.

Table 2: Paper Linking performance.

Method	Precision	Recall	F1-score
LSH	**98.72**	41.78	58.71
CNN-	90.84	89.45	90.14
CNN	96.60	94.55	95.57
LSH+CNN	96.70	**94.70**	**95.69**

Table 3: Running time of different methods for paper linking (in second).

Method	Train	Test	Predict
Keyword	-	41.82	41.24
SVM	30.07	31.62	30.94
Dedupe	1.5hrs+	1109.08	1108.94
LinKG$_L$	369.39	1545.24	1496.04
LSH	461.70	**0.56**	**0.16**
CNN	431.40	164.37	162.32
LSH+CNN	893.10	112.86	108.06

5.3 Detailed Result Analysis

Model Variants of Venue Linking. In Table 1, LinKG$_L$ uses two venue attributes as input, including integral sequences and keyword sequences. We also examine the result of only using integral sequence without additional features, and F1-score is 84.31%. After adding keyword sequences, F1-score can reach 86.76%. It shows that extracting keyword sequence plays an important role in venue linking. By further integrating two additional statistical features, including the number of reverse pairs, Jaccard Index of two venue names, the final F1-score reaches 89.33%.

Model Variants of Paper Linking. The variants of our model for paper linking are denoted as follows. (1) the method using locality-sensative hashing is denoted as LSH. (2) methods using CNN: The method only using paper titles to build similarity matrix is denoted as CNN-. Different from CNN-, CNN considers two paper attributes: titles and authors as described in Sec. 4.2. (3) hybrid model: LSH+CNN is a hybrid model of LSH and CNN. It first uses LSH to find matched papers efficiently. For those papers which can not be matched by binary codes, it utilizes CNN for fine-grained matching.

Table 2 shows the linking results of our model variants. In general, CNN-based methods outperform LSH significantly. LSH can achieve high precision. One possible reason is that the same title will be definitely mapped to the same binary code. Thus, under these circumstances, using hashing can avoid unnecessary computational cost, but will sacrifice recall. For hybrid method LSH+CNN, LSH can speed up the matching process and keep high precision and recall at the same time.

Efficiency Performance on Paper Linking. Furthermore, we test the efficiency of different methods for paper linking as shown in Table 3. We measure three types of running time: training time, testing time and predicting time. Testing time refers to the total time for test while predicting time focuses on core matching process which ignores some data processing steps since these steps are not the bottleneck of our methods. Predicting time sometimes is longer than training time because each paper needs to compare itself with all candidate pairs. Keyword method doesn't need training and the main time cost lies in string similarity calculation. LSH consumes the shortest predicting time because the complexity of matching is $O(n)$ and binary codes query can be extremely fast. Therefore, LSH is potential to handle large-scale data in an efficient way. The training time of hashing-based methods refers to the time of training Doc2Vec model. In spite of high accuracy, CNN-based method requires constructing similarity matrices and training convolutional neural networks so its computational cost is high. LSH+CNN can leverage LSH to match about 40% papers fast, so it reduces nearly half of the time compared with CNN. Time cost of Dedupe is also high because its blocking strategy takes a lot of training time.

Table 4: Author Linking results of our model variants.

Method	Precision	Recall	F1-score
SVM-struct	89.12	**96.17**	92.51
GAT	92.26	93.28	92.77
HGAT	93.39	93.96	93.67
HGAT-esb1	93.70	94.75	94.22
HGAT-esb2	**95.37**	93.48	**94.42**

Contribution Analysis for HGAT on Author Linking. The results in Table 4 show the contribution of components of our author linking model. GAT means directly applying original GAT on paired subgraphs and HGAT refers to using different attention mechanisms in GAT for different types of entities. SVM-struct refers to using some structural statistical features, including venue/paper matching ratio of two authors. HGAT-esb1 and HGAT-esb2 are two ensemble methods which add features used in SVM-struct (denoted as f_{stat}) to HGAT. Specifically, HGAT-esb1 concatenates f_{stat} and output vector h_{enc} of the encoder in HGAT as the final feature vector f_{cat}, and uses the objective of HGAT to train. HGAT-esb2 uses SVM as the classifier and f_{cat} as input features.

As shown in Table 4, using different attention parameters for different entities is better than treating them equally (0.9% F1-score improvement). Furthermore, the result of HGAT-esb2 is a little bit better than HGAT-esb1, which shows SVM does better here in processing combined features of statistical features and features generated by neural networks. Also, SVM-struct performs well (close to pure GAT), which demonstrates the effectiveness of heterogeneous structure information (venue/paper linking results) for author linking problem.

5.4 Open Academic Graph (OAG)

Based on our proposed framework, we have published Open Academic Graph (OAG)[1] , which is the largest publicly available heterogeneous academic graph as far as we know.

In OAG, we generated 91,137,597 paper linking pairs and its accuracy is 99.10%. We also successfully linked 29,841 venue pairs with accuracy of 99.26%. For authors, we only considered authors who published more than five papers since both AMiner and MAG are faced with the under-conflation and the over-conflation problem for author profiles. Finally, there were 6,855,193 authors in AMiner and 13,173,936 authors in MAG. We generated 1,717,680

author pairs and the estimated accuracy is 97.41%. The evaluation was based on a subset of sampled matchings (around one thousand venue/paper/author pairs). OAG has considerable applications. It can be used as a benchmark for studying citation networks, author name disambiguation, paper content, as well as comparing methodologies for data integration.

6 CONCLUSION

In this paper, we study an important problem of linking large-scale heterogeneous entity graphs. We particularly focus on building a large linked academic entity graph. We propose a unified framework, LinKG, to deal with the linking problem. LinKG is coupled with three linking modules, each of which addresses one category of entities. We evaluated the proposed framework and compared it with several state-of-the-art approaches. Experimental results show that our proposed framework LinKG can achieve a very high linking accuracy with a F1-score of 0.9510, significantly outperforming the states-of-the-arts. LinKG has been deployed to Microsoft Academic Search and AMiner to integrate the two large sources. The linked results have been published as Open Academic Graph (OAG).

ACKNOWLEDGMENTS

Jie Tang is the corresponding author of this paper. The work is supported by the NSFC for Distinguished Young Scholar (61825602) and NSFC (61836013), and a research fund supported by MSRA. Xiao Liu is supported by Tsinghua University Initiative Scientific Research Program and DCST Student Academic Training Program.

REFERENCES

[1] Ron Bekkerman and Andrew McCallum. 2005. Disambiguating Web Appearances of People in a Social Network. In *WWW'05*. 463–470.
[2] Omar Benjelloun, Hector Garcia-Molina, David Menestrina, Qi Su, Steven Euijong Whang, and Jennifer Widom. 2009. Swoosh: a generic approach to entity resolution. *The VLDB Journal* 18, 1 (2009), 255–276.
[3] Indrajit Bhattacharya and Lise Getoor. 2007. Collective entity resolution in relational data. *TKDE* 1, 1 (2007), 5.
[4] Mikhail Bilenko. 2004. Learnable Similarity Functions and their Applications to Clustering and Record Linkage. In *AAAI'04*. 981–982.
[5] George E Dahl, Tara N Sainath, and Geoffrey E Hinton. 2013. Improving deep neural networks for LVCSR using rectified linear units and dropout. In *ICASSP'13*. 8609–8613.
[6] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. 2004. Locality-sensitive hashing scheme based on p-stable distributions. In *SCG'04*. 253–262.
[7] Ahmed K Elmagarmid, Panagiotis G Ipeirotis, and Vassilios S Verykios. 2007. Duplicate record detection: A survey. *TKDE* 19, 1 (2007), 1–16.
[8] Hui Han, Lee Giles, Hongyuan Zha, Cheng Li, and Kostas Tsioutsiouliklis. 2004. Two Supervised Learning Approaches for Name Disambiguation in Author Citations. In *JCDL'04*. 296–305.
[9] Mark Heimann, Haoming Shen, Tara Safavi, and Danai Koutra. 2018. Regal: Representation learning-based graph alignment. In *CIKM'18*. 117–126.
[10] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.
[11] Lili Jiang, Jianyong Wang, Ning An, Shengyuan Wang, Jian Zhan, and Lian Li. 2009. Grape: A graph-based framework for disambiguating people appearances in web search. In *ICDM'09*. 199–208.
[12] Mahboob Alam Khalid, Valentin Jijkoun, and Maarten De Rijke. 2008. The impact of named entity normalization on information retrieval for question answering. In *ECIR'08*. 705–710.
[13] Quoc V Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents.. In *ICML*, Vol. 14. 1188–1196.
[14] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.
[15] Lingli Li, Jianzhong Li, and Hong Gao. 2015. Rule-Based method for entity resolution. *TKDE* 27, 1 (2015), 250–263.
[16] Xin Li, Paul Morie, and Dan Roth. 2004. Identification and tracing of ambiguous names: Discriminative and generative approaches. In *AAAI'04*. 419–424.
[17] Li Liu, William K Cheung, Xin Li, and Lejian Liao. 2016. Aligning Users across Social Networks Using Network Embedding. In *IJCAI*. 1774–1780.
[18] Tong Man, Huawei Shen, Shenghua Liu, Xiaolong Jin, and Xueqi Cheng. 2016. Predict Anchor Links across Social Networks via an Embedding Approach.. In *IJCAI'16*. 1823–1829.
[19] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS'13*. 3111–3119.
[20] Andrea Moro, Alessandro Raganato, and Roberto Navigli. 2014. Entity linking meets word sense disambiguation: a unified approach. *Transactions of the Association for Computational Linguistics* 2 (2014), 231–244.
[21] Jiezhong Qiu, Jian Tang, Hao Ma, Yuxiao Dong, Kuansan Wang, and Jie Tang. 2018. Deepinf: Social influence prediction with deep learning. In *KDD'18*. ACM, 2110–2119.
[22] Shu Rong, Xing Niu, Evan Xiang, Haofen Wang, Qiang Yang, and Yong Yu. 2012. A machine learning approach for instance matching based on similarity metrics. In *ISWC'12*. 460–475.
[23] Sunita Sarawagi and Anuradha Bhamidipaty. 2002. Interactive deduplication using active learning. In *KDD'02*. 269–278.
[24] Wei Shen, Jiawei Han, and Jianyong Wang. 2014. A probabilistic model for linking named entities in web text with heterogeneous information networks. In *SIGMOD'14*. ACM, 1199–1210.
[25] Wei Shen, Jianyong Wang, and Jiawei Han. 2015. Entity linking with a knowledge base: Issues, techniques, and solutions. *TKDE* 27, 2 (2015), 443–460.
[26] Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, Bo-june Paul Hsu, and Kuansan Wang. 2015. An overview of microsoft academic service (mas) and applications. In *WWW'15*. 243–246.
[27] Rebecca C. Steorts, Samuel L. Ventura, Mauricio Sadinle, and Stephen E. Fienberg. 2014. A Comparison of Blocking Methods for Record Linkage. In *PSD'14*. 253–268.
[28] Yizhou Sun and Jiawei Han. 2013. Mining heterogeneous information networks: a structural analysis approach. *Acm Sigkdd Explorations Newsletter* 14, 2 (2013), 20–28.
[29] Shulong Tan, Ziyu Guan, Deng Cai, Xuzhen Qin, Jiajun Bu, and Chun Chen. 2014. Mapping Users across Networks by Manifold Alignment on Hypergraph. In *AAAI'14*. 159–165.
[30] Jie Tang, A.C.M. Fong, Bo Wang, and Jing Zhang. 2012. A Unified Probabilistic Framework for Name Disambiguation in Digital Library. *IEEE TKDE* 24, 6 (2012), 975–987.
[31] Jie Tang, Juanzi Li, Bangyong Liang, Xiaotong Huang, Yi Li, and Kehong Wang. 2006. Using Bayesian decision for ontology mapping. *Journal of Web Semantics* 4, 4 (2006), 243–262.
[32] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *WWW'15*. 1067–1077.
[33] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. 2008. ArnetMiner: Extraction and Mining of Academic Social Networks. In *KDD'08*. 990–998.
[34] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. *ICLR* (2018).
[35] Zhichun Wang, Juanzi Li, Zhigang Wang, and Jie Tang. 2012. Cross-lingual knowledge linking across wiki knowledge bases. In *WWW'12*. 459–468.
[36] Yang Yang, Yizhou Sun, Jie Tang, Bo Ma, and Juanzi Li. 2015. Entity Matching Across Heterogeneous Sources. In *KDD'15*. 1395–1404.
[37] Xiaoxin Yin, Jiawei Han, and Philip S Yu. 2007. Object distinction: Distinguishing objects with identical names. In *ICDE'07*. 1242–1246.
[38] Jing Zhang, Bo Chen, Xianming Wang, Hong Chen, Cuiping Li, Fengmei Jin, Guojie Song, and Yutao Zhang. 2018. MEgo2Vec: Embedding Matched Ego Networks for User Alignment Across Social Networks. In *CIKM'18*. 327–336.
[39] Yutao Zhang, Jie Tang, Zhilin Yang, Jian Pei, and Philip Yu. 2015. COSNET: Connecting Heterogeneous Social Networks with Local and Global Consistency. In *KDD'15*. 1485–1494.
[40] Yutao Zhang, Fanjin Zhang, Peiran Yao, and Jie Tang. 2018. Name Disambiguation in AMiner: Clustering, Maintenance, and Human in the Loop.. In *KDD'18*. 1002–1011.
[41] Yan Zhuang, Guoliang Li, Zhuojian Zhong, and Jianhua Feng. 2017. Hike: A hybrid human-machine method for entity alignment in large-scale knowledge bases. In *CIKM'17*. 1917–1926.

A APPENDICES

A.1 Implementation Notes

A.1.1 LSTM for venue linking. In LinKG experiment, we build the LSTM model using Python package Keras 2.2.4. Empirically, we set the following parameters: training epochs=20, batch size=32.

We set the maximum length of integral/keyword sequence as 17/8. The embedding size of word in all sequences is 128. We use 2 LSTM layers: hidden layer size=32, dropout=0.2 and 3 Dense layers: hidden size = 64, 16, 1

Furthermore, we stack the jaccard index 32 times, number of inversion pairs 16 times to become the input vector JA_{ij} and IV_{ij}.This aims at enhancing their weights in the network.

We employ cross-entropy as our loss function and nadam as optimizer. All codes are implemented in Python3 and run by intepreter Python3.5. The experiments were conducted on a CentOS server with two Intel Xeon(R) CPU E5-4650 (2.7GHz) and 500G RAM.

• Code: https://github.com/zfjsail/OAG/

A.1.2 CNN for Paper Linking. Our CNN model is implemented in Python package Tflearn 0.3.2. As is demonstrated in 4.2, we leverage title and author information as input in this CNN model. Empirically, we set the parameters as follows:

(1) CNN network for title: activation function=Relu,
   - CNN layer 1: input size=(7,7,1), input channel=1, output channel=8, kernel size=3, regularizer=L2
   - CNN layer 2: input size=(5,5,8), input channel=8, output channel=16, kernel size=2, regularizer=L2
   - Fully-connected layer: output size=512
(2) CNN network for author: activation function=Relu
   - CNN layer: input size=(4,4,1), input channel=1, output channel=8, kernel size=2
   - Fully-connected layer: output size=512
(3) Fully-connected layer: output size=2, activation function= softmax, dropout = 0.5

The whole network uses learning rate=0.02, Adagrad optimizer, categorical cross entropy as loss function. The experiments were conducted on a Ubuntu server with four Intel Xeon(R) CPU E5-2699 (2.7GHz), 256G RAM and an Nvidia Titan X(Pascal) GPU and 12G GPU RAM.

• Code: https://github.com/zfjsail/OAG/

A.1.3 LSH for Paper Linking. The dimension of Doc2Vec embedding is set as 100 and the dimension of paper binary codes as 128. The experiment was conducted on a Ubuntu server with four Intel Xeon(R) CPU E5-2699 (2.20GHz) and 256 RAM.

• Code: https://github.com/zfjsail/OAG/

A.1.4 HGAT for Author Linking. Empirically, we set the initialization settings as: random seed=42, training epochs=30, weight decay(L2 loss on parameters)=1e-3, dropout=0.2, attn dropout=0, batch size=64

In terms of Graph Attention networks, we employ 2 multihead graph attention layer in it. Their parameters are as follows:

(1) Multihead graph attention layer 1: input size=1433, output size=8, n_head=8
(2) Multihead graph attention layer 2: input size=64, output =7, n_head=1
(3) Fully-connected layer 1: input length=7, output length=21
(4) Fully-connected layer 2: input length=21, output length=7
(5) Fully-connected layer 3: input length=7, output length=2

We employ Adam as our optimizer. All the HGAT codes are implemented in Python3 package Pytorch 1.0.0, and the experiment was conducted on a Ubuntu server with four Intel Xeon(R) CPU E5-2699 (2.7GHz), 256G RAM and an Nvidia Titan X(Pascal) GPU and 12G GPU RAM.

• Code: https://github.com/zfjsail/OAG/. This part of the code refers to the implementation in [21].

A.2 Baselines

A.2.1 Keywords.

- **For venues:** we use Python package Scikit-learn 0.19.1 to build word-frequency matrix and calculate tf-idf value for each venue name. The corpus is consisted of all the appeared words in the training and testing datasets. Then we leverage Cosine Similarity implemented in numpy by ourselves to calculate the pairwise similarity.
- **For papers:** in CP, we use the first γ words in paper titles to build inverted index table. Given a paper $p \in NP$, we use its titles to find candidate papers in CP by inverted index table and then re-rank these candidates by edit distance between titles and authors' similarity. For authors' similarity, we concatenate authors' name as a string and then calculate similarity in terms of character co-occurrence, i.e. $SIM_{authors}(a_1, a_2) = \frac{chars\_co\_occur(a_1, a_2)}{\max(\#chars(a_1), \#chars(a_2))}$
- **For authors:** we simply compare the two name strings to match authors.

A.2.2 SVM. We use Python package Scikit-learn 0.19.1 to build the SVM model on a macOS 10.14.2 laptop with an Intel Core i5 (2.9GHz) and 16GB RAM.

- **For venues:** we calculate the Jaccard Index and Cosine Similarity (using tf-idf values) for each pair, and let them be the two dimensions of the input vector.
- **For papers:** we calculate the character-level n-gram similarity of paper title (n=4), and authors (n=3).
- **For authors:** we calculate the character-level n-gram similarity (here we set n=4) of name, affiliation, top venue name, paper title keywords, and top coauthor name from each candidate pair. Note that paper title keywords are extracted from the first 15 papers in the paperlist of an author. MAG papers are replaced by their matched AMiner papers.

A.2.3 Dedupe. We use Open Source Dedupe 1.9.4 from Github. We employ the bottom layer interface *dedupe.core.scoreDuplicates* to obtain pairwise similarity scores between entities.

- **For venues:** we simply pass names of training pairs to Dedupe, and use the model to predict on test pairs.
- **For papers:** we pass paper's title, authors' name, venue, year as a dict to Dedupe.
- **For authors:** we pass author's name, affiliation, top venue name, and top coauthor name as a dict to Dedupe.

The experiment was conducted on a CentOS server with two Intel Xeon(R) CPU E5-4650 (2.7GHz) and 500G RAM.

• Code: https://github.com/dedupeio/dedupe

A.2.4 COSNET [39] . We download the authors' source codes, which were written in C++. The author provides 4 samples and 5 input options to choose. We maintain the original settings as before, and examine the 4 samples. Finally we choose to generate data in the form of $imdb\_refined.dat$, which contains both the pairwise shared keywords and relations between candidate pairs.

In terms of the 5 options: (1) training dataset: here we choose our own datasest. (2) labeling rate [0-5]: the higher rate is, the better training result will be. (3) reservoir size [0-4]. (4) query method [0-3]. (5) sampling method [0-3].

After tuning, we find the combination of labeling rate[5], reservoir size[4], query method[0], sampling method[0] could reach the best testing F1, and we report it in our experiment section.

This experiment was conducted on a macOS 10.14.2 laptop with an Intel Core i5 (2.9GHz) and 16GB RAM. The C++ compiler is clang-1000.11.45.5.

• Code: https://www.aminer.cn/cosnet

A.2.5 MEgo2Vec [38]. We download the original codes, and keep all the training settings as the same. MEgo2Vec needs two parts of input: one is attributes of a candidate pair, another is probably correct neighbor pair of a candidate pair, which is determined by name matching.

Therefore, on one hand, we pass author's name, affiliation, top venue name and top coauthor name to MEgo2Vec. On the other hand, we perform a rough name matching around the candidate pair, and extract neighbor pairs.

We make some changes in codes to better evaluate this model:

(1) The codes were originally written in Python2. We fix a few grammar mistakes and turn codes into Python3.
(2) There is no test part in the original codes, and we copy the validation part in the original codes and complete the test part.

We run the model on a CentOS server with two Intel Xeon(R) CPU E5-4650 (2.7GHz) and 500G RAM.

• Code: https://github.com/BoChen-Daniel/MEgo2Vec-Embedding-Matched-Ego-Networks-for-User-Alignment-Across-Social-Networks

A.3 Dataset

A.3.1 Venue Dataset. Initially, we performed a rough name matching using Jaccard Index with threshold of 0.4 on the original AMiner and MAG venue sets, generating 3344 pairs(1557 MAG venues, 3344 AMiner venues). We randomly selected 548 MAG venues, and generated 1202 candidate pairs for training. We labeled them manually as our ground truths.

This Venue Dataset is difficult, because candidate pairs with the same MAG venue often share a number of keywords, leading to high similarity in most of text similarity methods. This dataset will be published with our model codes.

A.3.2 Paper Dataset. we create two paper sets: clean paper sets CP and noisy paper sets NP. The first dataset is collected by extracting 46,170 papers from AMiner. Each paper contains 4 attributes: title, authors, venue and year. Another dataset is constructed by adding noise to each paper. Thus, each original paper and its corresponding noisy paper become a matching pair, which avoids the labeling cost of human labors. The method to add noise is based on our statistics of differences in existing matched papers. For example, some words in title are wrongly concatenated and authors' names have different formats, such as full name and abbreviated name. The two paper sets form one-to-one positive matching pairs. The negative paper pairs we generate share some common keywords to increase linking difficulty.

A.3.3 Author Dataset. At first, we conducted very strict rule methods on AMiner and MAG authors datasets to generate ground-truth linkings, including affiliation matching rate, venue matching rate and paper matching rate. Furthermore, we utilize the generated linked authors' coauthors to match more authors.

Since our groundtruth is generated by rules, it might be a little bit easy for Author Linking task. Therefore, we implemented the following two methods to harden this dataset:

(1) Add name ambiguity: for an already linked author entity, we searched for authors with the same name, and generated negative candidate pairs.
(2) Add noisy data: we randomly replace some attributes of generated positive pairs, including affiliations, venues and papers.

This author dataset contains 10,000 positive candidate pairs and 10,000 negative candidate pairs. It will also be published with our model codes.

A.4 Discussions

It's difficult to obtain a "ground truth" for entity linking evaluation. In this paper, we manually label venue training data and construct artificial difficult training data for papers and venues. In the future, more ground truths may be obtained via crowd-sourced data management.

In this work, we first link relatively easy entities (i.e. venues and papers) and then link authors with more ambiguity. How to link large-scale different types of entities in a joint framework is also a challenge.

Furthermore, for author linking, we only consider authors with not less than 5 papers for both sides. This is because, due to the name ambiguity problem, both AMiner and MAG are facing with the under-conflation and the over-conflation problem for author profiles. Therefore, how to improve author name disambiguation performance by leveraging current disambiguation results and linked entities is also an important challenge.