A PSEUDOCODE OF REA TRAINING PROCEDURE

The pseudocode of REA training procedure is described in Algorithm 1. After training, the optimized embeddings $\mathbf{E}^i$ and $\mathbf{E}^j$ can be utilized to do the inference.

Algorithm 1: Model training for REA

Input: KG $\mathcal{G}_i$ and $\mathcal{G}_j$, the set of labeled entity pairs $\mathcal{AS}^U$, the set of trusted entity pairs $\mathcal{AS}^T$.
Output: Embeddings of entities in $\mathcal{G}_i$ and $\mathcal{G}_j$.

Initialize embeddings $\mathbf{E}^i$, $\mathbf{E}^j$, and parameters of G, D and f_{enc};
Initialize the trust scores for $\mathcal{AS}^U$ and $\mathcal{AS}^T$;
while *not converge* **do**
  `// Noise-aware entity alignment training`
  for $m = 0; m < m_{EN}$ **do**
    Sample a batch of entity pairs from $\mathcal{AS}^T$;
    Update $\mathcal{L}_{EA}$ according to Eq. 6
  end
  `// Noise discriminator training`
  for $m = 0; m < m_D$ **do**
    Sample a batch of entity pairs in $\mathcal{AS}^T$, *i.e.*, (e_x, e_y);
    Generate m_s noisy entity pairs $(e'_x, e'_y) \sim G(\cdot|(e_x, e_y))$ for each (e_x, e_y) ;
    Update φ according to Eq. 15 using (e_x, e_y) and (e'_x, e'_y);
  end
  `// Noisy entity pair generator training`
  for $m = 0; m < m_G$ **do**
    Sample a batch of entity pairs in $\mathcal{AS}^T$, *i.e.*, (e_x, e_y);
    Generate m_s noisy entity pairs $(e'_x, e'_y) \sim G(\cdot|(e_x, e_y))$ for each (e_x, e_y) ;
    Update θ according to Eq. 10 using (e_x, e_y) and (e'_x, e'_y);
  end
  Update the trust scores of entity pairs in $\mathcal{AS}^U$;
  Add entity pairs in $\mathcal{AS}^U$ with TS = 1 into $\mathcal{AS}^T$;
end

B EXPERIMENTAL SETTINGS.

B.1 Environment

We implement the proposed framework using the Python library Tensorflow and conduct all the experiments on Linux server with GPUs (GeForce RTX 2080 Ti) and CPU (Intel Xeon). We will release the source code of REA at https://github.com/scpei/REA. And the public datasets used in the paper are available at: http://ws.nju.edu.cn/jape/.

B.2 Parameter Settings

In this work, we adopt widely-used evaluation metrics, Hits@K and MRR for evaluating entity alignment results as existing works did. We set the dimension of entity embedding as 200 for all methods. For baselines, we find the optimal parameters or follow the settings in original papers to achieve the best performance. For our REA method, we apply a grid search for hyper-parameters and find the best configuration: margin hyper-parameter γ is 3.0, the number of GCN layers is 2, and MLPs used in generator and discriminator have two-layers with 100 and 30 hidden units, respectively. We set m_{EN} as 1000, m_D and m_G as 1000. In addition, we set the number of negative samples k as 10. Moreover, we set trust scores as binary value, and threshold δ as 0.01, in order to achieve a better balance between the performance of entity alignment and the impact of noise. We use Adam optimizer to optimize the loss function in Eq. 15 and Eq. 10 with learning rate 0.01, and apply SGD to optimize the loss function in Eq. 6. Each evaluation is repeated 5 times and averaged results are reported.