A Critical Assessment of State-of-the-Art in Entity Alignment

Max Berrendorf[1 [0000−0001−9724−4009]], Ludwig Wacker[1], and Evgeniy Faerman[1]

Ludwig-Maximilians-Universität München, Munich, Germany
{berrendorf,faerman}@dbs.ifi.lmu.de, l.wacker@campus.lmu.de

Abstract. In this work, we perform an extensive investigation of two state-of-the-art (SotA) methods for the task of Entity Alignment in Knowledge Graphs. Therefore, we first carefully examine the benchmarking process and identify several shortcomings, which make the results reported in the original works not always comparable. Furthermore, we suspect that it is a common practice in the community to make the hyperparameter optimization directly on a test set, reducing the informative value of reported performance. Thus, we select a representative sample of benchmarking datasets and describe their properties. We also examine different initializations for entity representations since they are a decisive factor for model performance. Furthermore, we use a shared train/validation/test split for a fair evaluation setting in which we evaluate all methods on all datasets. In our evaluation, we make several interesting findings. While we observe that most of the time SotA approaches perform better than baselines, they have difficulties when the dataset contains noise, which is the case in most real-life applications. Moreover, we find out in our ablation study that often different features of SotA method are crucial for good performance than previously assumed. The code is available at https://github.com/mberr/ea-sota-comparison.

Keywords: Knowledge Graph · Entity Alignment · Word Embeddings

1 Introduction

The quality of information retrieval crucially depends on the accessible storage of information. Knowledge Graphs (KGs) often serve as such data structure [6]. Moreover, to satisfy diverse information needs, a combination of multiple data sources is often inevitable. Entity Alignment (EA) [2] is the discipline of aligning entities from different KGs. Once aligned, these entities facilitate information transfer between both knowledge bases, or even fusing multiple KGs to a single knowledge base.

In this work, our goal is to analyze a SotA approach for the task of EA and to identify which factors are essential for its performance. Although papers often use the same dataset in the evaluation and report the same evaluation metrics, the selection of SotA is not a trivial task: As we found out in our analysis, the usage of different types of external information for the initialization or train/test splits

of diffing sizes[1] make the results in different works incomparable. Therefore, while still guided by the reported evaluation metrics, we identified the following factors common among strongly performing methods in multiple works:

- They are based on Graph Neural Networks (GNNs). GNNs build the basis of the most recent works [16,9,12,21,22,4,23,25,10,14,7,17,20,18,19].
- The utilize entity names in the model. Supported by recent advances in word embeddings, these attributes provide distinctive features.
- They consider different types of relations existing in KGs. Most GNNs ignore different relationship types and aggregate them in the preprocessing step.

Given these criteria, we selected RDGCN [17], as it also has demonstrated impressive performance in recent benchmarking studies [15,24]. Additionally, we include the recently published DGMC [7] method in our analysis for two reasons: The studies mentioned above did not include it, and the authors reported surprisingly good performance, considering that this method does not make use of relation type information.

We start our study by reviewing the used datasets and discussing the initializations based on entity names. Although both methods utilize entity names, the actual usage differs. For comparison, we thus evaluate both methods on all datasets with all available initializations. We also report the zero-shot performance, i.e., when only using initial representations alone, as well as a simple GNN model baseline. Furthermore, we address the problem of hyperparameter optimization. Related works often do not discuss how they chose hyperparameters, and e.g. rarely report validation splits.[2] Also, in the published code of the investigated approaches, we could not find any trace of train-validation splits, raising questions about reproducibility and fairness of their comparisons. We thus create a shared split with a test, train, and validation part and extensively tune the model's hyperparameters for each of the dataset/initialization combinations to ensure that they are sufficiently optimized. Finally, we provide an ablation study for many of the parameters of a SotA approach (RDGCN), giving insight into the individual components' contributions to the final performance.

2 Datasets & Initialization

Table 1 provides a summary of a representative sample of datasets used for benchmarking of EA approaches. In the following, we first discuss the properties of each dataset and, in the second part, the initialization of entity name attributes.

[1] Commonly used evaluation metrics in EA automatically become better with a smaller size of test set [3].

[2] [15] use cross-validation, but only for training the model's parameters. The hyperparameter choice is still based on published parameters or manually chosen.

Table 1. Summary of the used EA datasets. We denote the entity set as $\mathcal{E}$, the relation set as $\mathcal{R}$, the triple set as $\mathcal{T}$, the aligned entities as $\mathcal{A}$ and the exclusive entities as $\mathcal{X}$.

dataset	subset	graph	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{R}\rvert$	$\lvert\mathcal{T}\rvert$	$\lvert\mathcal{A}\rvert$	$\lvert\mathcal{X}\rvert$
dbp15k	zh-en	zh	19,388	1,701	70,414	15,000	4,388
		en	19,572	1,323	95,142	15,000	4,572
	ja-en	ja	19,814	1,299	77,214	15,000	4,814
		en	19,780	1,153	93,484	15,000	4,780
	fr-en	fr	19,661	903	105,998	15,000	4,661
		en	19,993	1,208	115,722	15,000	4,993
wk3l15k	en-de	en	15,126	1,841	209,041	9,783	5,343
		de	14,603	596	144,244	10,021	4,582
	en-fr	en	15,169	2,228	203,356	7,375	7,794
		fr	15,393	2,422	169,329	7,284	8,109
openea	en-de	en	15,000	169	84,867	15,000	0
		de	15,000	96	92,632	15,000	0
	en-fr	en	15,000	193	96,318	15,000	0
		fr	15,000	166	80,112	15,000	0
	d-y	d	15,000	72	68,063	15,000	0
		y	15,000	21	60,970	15,000	0
	d-w	d	15,000	167	73,983	15,000	0
		w	15,000	121	83,365	15,000	0

2.1 Datasets

DBP15k The DBP15k dataset is the most popular dataset for the evaluation of EA approaches. It has three subsets, all of which base upon DPedia, and comprise a pair of graphs from different languages. As noted by [2], there exist multiple variations of the dataset, sharing the same entity alignment, but differing in the number of exclusive entities in each graph. The alignments in the datasets are always 1:1 alignments, and due to the construction method for the datasets, exclusive entities do not have relations between them, but only to shared entities. Exclusive entities complicate the matching process and in real-life application, they are not easy to identify. Therefore, we believe that this dataset describes a realistic use-case only to a certain extent. We found another different variant of DBP15k as part of the PyTorch Geometric repository[3], having a different set of aligned entities. This is likely due to extraction of alignments from data provided by [20] via Google Drive[4] as described in their GitHub repository.[5] As a result, the evaluation results published in [7] are not directly comparable to other published results. In our experiments, we use the (smaller) JAPE variant with approximately 19-20k entities in each graph, since it is the predominantly used variant.

OpenEA The OpenEA datasets published by [15] comprise graph pairs from DBPedia, YAGO, and Wikidata obtained by iterative degree-based sampling to match the degree distribution between the source KG and the extracted subset. The aligments are exclusively 1:1 matchings, and there are no exclusive entities,

[3] https://github.com/rusty1s/pytorch_geometric/blob/d42a690fba68005f5738008a04f375ffd39bbb76/torch_geometric/d
[4] https://drive.google.com/open?id=1dYJtj1_J4nYJdrDY95ucGLCuZXDXI7PL
[5] https://github.com/syxu828/Crosslingula-KG-Matching/blob/56710f8131ae072f00de97eb737315e4ac9510f2/READM

i.e., every entity occurs in both graphs. We believe that this is a relatively unrealistic scenario. In our experiments, we use all graph pairs with 15k entities (`15K`) in the dense variant (`V2`), i.e., `en-de-15k-v2`, `en-fr-15k-v2`, `d-y-15k-v2`, `d-w-15k-v2`.

WK3l15k The Wk3l datasets are multi-lingual KG pairs extracted from Wikipedia. As in [2], we extract additional entity alignments from the triple alignments. The graphs contain additional exclusive entities, and there are m:n matchings. We only use the 15k variants, where each graph has approximately 15k entities. There are two graph pairs, `en-de` and `en-fr`. Moreover, the alignments in the dataset are relatively noisy: For example, `en-de` contains besides valid alignments such as ("trieste", "triest"), or ("frederick i, holy roman emperor", "friedrich i. (hrr)"), also ambiguous ones such as ("1", "1. fc saarbrücken"), ("1", "1. fc schweinfurt 05"), and errors such as ("1", "157"), and ("101", "100"). While the noise aggravates alignment, it also reflects a realistic setting.

2.2 Label-Based Initializations

Prepared translations (DBP15k) For DBP15k, we investigate label-based initializations based on prepared translations to English from [17] and [7] (which, in turn, originate from [20]). Afterwards, they use Glove [11] embeddings to obtain an entity representation. While [17] only provides the final entity representation vectors without further describing the aggregation, [7] splits the label into words (by white-space) and uses the sum over the words' embeddings as entity representation. [17] additionally normalizes the norm of the representations to unit length.

Table 2. Statistic about label-based initialization in the OpenEA codebase: *attribute* denotes initialization via attribute values for a predefined set of "name attributes". *id* denotes initialization with the last part of the entity URI. For `d-y` this basically leaks ground truth, whereas, for Wikidata, the URI contains only a numeric identifier, thus rendering the initialization "label" useless.

subset	side	via attribute	via id	via id (%)
d-w	d	0	15,000	100.00%
	w	8,391	7,301	48.67%
d-y	d	2,883	12,122	80.81%
	y	15,000	0	0.00%

Prepared RDGCN Embeddings (OpenEA) OpenEA [15] benchmarks a large variety of contemporary entity alignment methods in a unified setting, also including RDGCN [17]. Since the graphs DBPedia and YAGO collect data from similar sources, the labels are usually equal. For those graph pairs, the authors propose to delete the labels. However, RDGCN requires a label based initialization.

Thus, the authors obtain labels via attribute triples of a pre-defined set of "name-attributes"[6]: `skos:prefLabel, http://dbpedia.org/ontology/birthName` for DBPedia-YAGO, and `http://www.wikidata.org/entity/P373, http://www.wikidata.org/entity/P1476` for DBPedia-Wikidata.

However, when investigating the published code we noticed that in case the label is not found via attribute, the last part of the entity URI is used instead. For DBPedia/YAGO this effectively leaks ground truth since they share the same label. For DBPedia/Wikidata, this results in useless labels for the Wikidata side since their labels are the Wikidata IDs, e.g., `Q3391163`. Table 2 summarizes the frequency of both cases. For `d-w`, DPBedia entities always use the ground truth label. For 49% of the Wikidata entities, useless labels are used for initialization. For `d-y`, YAGO entity representations are always initialized via an attribute triple. For DBPedia, in 81% of all cases, the ground truth label is used. We store these initial entity representations produced by the OpenEA codebase into a file and refer in the following to them as *Sun* initialization.

Multi-lingual BERT (WK3l) Since we did not find related work with entity embedding initialization from labels on WK3l15k, we generated those using a pre-trained multi-lingual BERT model [5], `BERT-Base, Multilingual Cased`[7]. Following [5], we use the sum of the last four layers as token representation since it has comparable performance to the concatenation at a quarter of its size. For summarization of the token representations of a single entity label, we explore sum, mean and max aggregation as hyperparameters.

3 Methods

We evaluate two SotA EA methods, RDGCN [17] which we reimplemented and DGMC [7] for which we used the original method implementation with adapted evaluation. In the following, we revisit their architectures and highlight differences between the architecture described in the paper and what we found in the published code.

Similarly to all GNN-based approaches, both models employ a Siamese architecture. Therefore, the same model with the same weights is applied to both graphs yielding representations of entities from both KGs. Given these entity representations, the EA approaches compute an affinity matrix that describes the similarity of entity representations from both graphs. Since the main difference between methods is the GNN model in the Siamese architecture, for brevity we only describe how it is applied on a single KG $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$.

3.1 Relation-aware Dual-Graph Convolutional Network (RDGCN)

Architecture The RDGCN [17] model comprises two parts performing message-passing processes applied sequentially. The message passing process performed

[6] https://github.com/nju-websoft/OpenEA/tree/2a6e0b03ec8cdcad4920704d1c38547a3ad72abe
[7] https://github.com/google-research/bert/blob/cc7051dc592802f501e8a6f71f8fb3cf9de95dc9/multilingual.md

by the first part can be seen as *relation-aware*. The model tries to learn the importance of relations and weighs the messages from the entities connected by these relations correspondingly. The message passing performed by the second component utilizes a simple adjacency matrix indicating the existence of any relations between entities, which we call *standard message passing*. Both components employ a form of skip connections: (weighted) residual connections [8] in the first part and highway layers [13] in the second part.

Relation-Aware Message Passing The entity embeddings from the first component are computed by several *interaction rounds* comprising four steps

$$\mathbf{X_c} = RC(\mathbf{X_e}), \mathbf{X_c} \in \mathbb{R}^{|\mathcal{R}| \times 2d} \tag{1}$$

$$\mathbf{X_r} = DA(\mathbf{X_r}, \mathbf{X_c}), \mathbf{X_r} \in \mathbb{R}^{|\mathcal{R}| \times 2d} \tag{2}$$

$$\mathbf{X_e} = PA(\mathbf{X_e}, \mathbf{X_r}) \tag{3}$$

$$\mathbf{X_e} = \mathbf{X_e^0} + \beta_i \cdot \mathbf{X_e} \tag{4}$$

The first step, in (1), obtains a *relation context* (RC) $\mathbf{X_c}$ from the entity representations. For relation $r \in \mathcal{R}$, we extract its relation context as a concatenation of the mean entity representations for the head and the tail entities. By denoting the set of head and tail entities for relation r with H_r and T_r, we can thus express its computation as $(\mathbf{X_c})_i = \left[{}^1\!/_{|H_i|} \sum_{j \in H_i} (\mathbf{X_e})_j \parallel {}^1\!/_{|T_i|} \sum_{j \in T_i} (\mathbf{X_e})_j \right]$ where $\parallel$ denotes the concatenation operation. An entity occurring multiple times as the head is weighted equally to an entity occurring only once.

The second step, in (2), is the *dual graph attention* (DA). The attention scores on the dual graph α_{ij}^D are computed by dot product attention with leaky ReLU activation: $\alpha_{ij}^D = J_{ij} \cdot LeakyReLU(\mathbf{W_L}(\mathbf{X_c})_i + \mathbf{W_R}(\mathbf{X_c})_j)$. Notice that $\mathbf{W_L}(\mathbf{X_c})_i + \mathbf{W_R}(\mathbf{X_c})_j = (\mathbf{W_L} \| \mathbf{W_R})^T ((\mathbf{X_c})_i \| (\mathbf{X_c})_j)$, where $\|$ denotes the concatenation operation. In the published code, we further found a weight sharing mechanism for $\mathbf{W_L}$ and $\mathbf{W_R}$ implemented, decomposing the projection weight matrices as $\mathbf{W_L} = \mathbf{W'_L W_C}$ and $\mathbf{W_R} = \mathbf{W'_R W_C}$ with $\mathbf{W'_L}, \mathbf{W'_R} \in \mathbb{R}^{1 \times h}, \mathbf{W_C} \in \mathbb{R}^{h \times 2d}$ being trainable parameters, and $\mathbf{W_C}$ shared between both projections. J_{ij} denotes a fixed triple-based relation similarity score computed as the sum of the Jaccard similarities of the head and tail entity set for relation r_i and r_j: $J_{ij} := {}^{|H_i \cap H_j|}\!/_{|H_i \cup H_j|} + {}^{|T_i \cap T_j|}\!/_{|T_i \cup T_j|}$. The softmax is then computed only over those relations, where $J_{ij} > 0$, i.e. which share at least one head or tail entity. In the implementation, this is implemented as dense attention with masking, i.e. setting $\alpha_{ij}^D = -\infty$ (or a very small value) for $J_{ij} = 0$. While this increases the required memory consumption to $\mathcal{O}(|\mathcal{R}|^2)$, the number of relations is usually small compared to the number of entities, cf. Table 1, and thus this poses no serious computational problem. With $\tilde{\alpha}_{ij}^D$ denoting the softmax output, the new relation representation finally is $(\mathbf{X_r})_i = ReLU\left(\sum_j \tilde{\alpha}_{ij}^D (\mathbf{X_r})_j\right)$.

In the third step, in (3), the entity representations are updated. To this end, a relation-specific scalar score is computed as $\alpha_i^r = LeakyReLU(\mathbf{W}\mathbf{X_r} + b)$ with trainable parameters W and b. Based upon the relation-specific scores, an

attention score between two entities e_i, e_j with at least one relation between them is given as $\alpha^P_{ij} = \sum_{r \in \mathcal{T}_{ij}} \alpha^r_i$. These scores are normalized with a sparse softmax over all $\{j \mid \exists r \in \mathcal{R} : (e_i, r, e_j) \in \mathcal{T}\}$: $\tilde{\alpha}^P_{ij} = \text{softmax}_{j'}(\alpha^P_{ij'})_j$. The final output of the primal attention is $(\mathbf{X_e})_j = ReLU(\sum_i \tilde{\alpha}_{ij}(\mathbf{X_e})_j)$.

The fourth step, in (4), applies a skip connection from the initial representations to the current entity representation. The weight β_i is pre-defined ($\beta_1 = 0.1$, $\beta_2 = 0.3$) and not trained.

Standard Message Passing The second part of the RDGCN consists of a sequence of GCN layers with highway layers. Each layer computes

$$\mathbf{X'_e} = ReLU(\mathbf{A}\mathbf{X_e}\mathbf{W}) \tag{5}$$

$$\beta = \sigma(\mathbf{W_g}\mathbf{X_e} + b_g) \tag{6}$$

$$\mathbf{X_e} = \beta \cdot \mathbf{X'_e} + (1 - \beta) \cdot \mathbf{X_e} \tag{7}$$

$\mathbf{A} \in \mathbb{R}^{|\mathcal{E}^L| \times |\mathcal{E}^L|}$ denotes the adjacency matrix of the primal graph. It is constructed by first creating an undirected, unweighted adjacency matrix where there is a connection between $e_i, e_j \in \mathcal{E}^L$ if there exists at least one triple $(e_i, r, e_j) \in \mathcal{T}^L$ for some relation $r \in \mathcal{R}^L$. Next, self-loops (e, e) are added for every entity $e \in \mathcal{E}^L$. Finally, the matrix is normalized by setting $\mathbf{A} = \mathbf{D}^{-1}\mathbf{A}\mathbf{D}^{-1}$ with $\mathbf{D}$ denoting the diagonal matrix of node degrees. When investigating the published code, we further found out that the weight matrix $\mathbf{W}$ is constrained to be a diagonal matrix, and initialized as an identity matrix.

Training Let $\mathbf{x}^L_i$ denote the final entity representation for $e^L_i \in \mathcal{E}^L$ and anologously $\mathbf{x}^R_j$ for $e^R_j \in \mathcal{E}^R$. RDGCN is trained with a margin-based loss formulation. RDGCN adopts a hard negative mining strategy, i.e., the set of negative examples for one pair is the top k most similar entities of one of the entities according to the similarity measure used for scoring. In [17], sim is the negative l_1 distance, $\gamma = 1$, $k = 10$, and the negative examples are updated every 10 epochs.

3.2 Deep Graph Matching Consensus (DGMC)

DGMC [7] also comprises two parts, which we name *enrichment* and *correspondence refinement*. The enrichment part is a sequence of GNN layers enriching the entity representations with information from their neighborhood. Each layer computes $\phi(\mathbf{X}) = ReLU(norm(\mathbf{A})\mathbf{X}\mathbf{W_1} + norm(\mathbf{A}^T)\mathbf{X}\mathbf{W_2} + \mathbf{X}\mathbf{W_3})$, where $\mathbf{A} \in \mathbb{R}^{|\mathcal{E}^L| \times |\mathcal{E}^L|}$ denotes the symmetrically normalized adjacency matrix (as for RDGCN), *norm* the row-wise normalization operation, $\mathbf{X} \in \mathbb{R}^{\mathcal{E}^L \times d_{in}}$ the layer's input, and $\mathbf{W_1}, \mathbf{W_2}, \mathbf{W_3} \in \mathbb{R}^{d_{in} \times d_{out}}$ trainable parameters of the layer. An optional batch normalization and dropout follow this layer. For the final output of the enrichment phase, all individual layer's outputs are concatenated before a learned final linear projection layer reduces the dimension to d_{out}.

The second phase, the *correspondence refinement*, first calculates the $k = 10$ most likely matches in the other graph for each entity as a sparse correspondence matrix $\mathbf{S} \in \mathbb{R}^{|\mathcal{E}^L| \times |\mathcal{E}^R|}$, normalized using softmax. Next, it generates random vectors for each entity $\mathbf{R} \in \mathbb{R}^{|\mathcal{E}^L| \times d_{rnd}}$ and sends these vectors to the probable matches via the softmax normalized sparse correspondence matrix, $\mathbf{S}^T\mathbf{R} \in \mathbb{R}^{|\mathcal{E}^R| \times d_{rnd}}$. A GNN layer ψ as in phase one distributes these vectors in the neighborhood of the nodes: $\mathbf{Y}^R = \psi(\mathbf{S}^T\mathbf{R})$. A two-layer MLP predicts an update for the correspondence matrix, given the difference between the representations $\mathbf{Y}^L$ and $\mathbf{Y}^R$. This procedure is repeated for a fixed number of refinement steps $L = 10$.

4 Experiments

Experimental Setup For the general evaluation setting and description of metrics, we refer to [3]. Here, we primarily use Hits@1, which measures the relative frequency of the correct entity being ranked at the first position. When investigating the published code of both, RDGCN [17][8] and DGMC [7][9], we did not find any code for tuning the parameters, nor a train-validation split. Also, the papers itself do not mention a train-validation split. Thus, it is unclear how they choose the hyperparameters, without having a test-leakage by directly optimizing for performance on the test set. To enable a fair comparison we thus decided to create a shared test-train-validation split used by all our experiments. Since DGMC already uses PyTorch we could use their published code and extend it with HPO code. RDGCN was re-implemented in PyTorch in our codebase. We use the official train-test split for all datasets, which reserves 70% of the alignments for testing. We split the remaining part into 80% train alignments and 20% validation alignments.

We continued by tuning numerous model parameters (cf. Table 3) of all models on each of the datasets in Table 1 and each of the available initializations described in Section 2.2 to obtain sufficiently well-tuned configurations. We used random search due to its higher sample efficiency compared to grid search [1]. We additionally evaluate a baseline, which uses the GNN variant from DGMC without the neighborhood consensus refinement, coined *GCN-Align** due to its close correspondence to [16], and also evaluate the zero-shot performance of the initial node features.

For each tested configuration, we perform early stopping on validation H@1, i.e., select the epoch according to the best validation H@1. Across all tested configurations for a model-dataset-initialization combination, we then choose the best configuration according to validation H@1 and report the test performance in Table 4. We do not report performance for training on train+validation with the final configuration due to space restrictions. We decided to report performance when trained only on the train set to ensure that other works have performance numbers for comparison when tuning their own models.

[8] https://github.com/StephanieWyt/RDGCN
[9] https://github.com/rusty1s/deep-graph-matching-consensus/

Table 3. Investigated hyperparameters for all methods. * denotes that these parameters share the same value range but were tuned independently.

Common	
parameter	choices
optimizer	Adam
similarity	{cos, dot, l1 (bound inverse), l1 (negative), l2 (bound inverse), l2 (negative)}
RDGCN	
parameter	choices
(entity embedding) normalization	{always-l2, initial-l2, never}
(number of) GCN layers	{0, 1, 2, 3}
(number of) interaction layers	{0, 1, 2, 3}
interaction weights	$\{0.1, 0.2, \ldots, 0.6\}$
trainable embeddings	{False, True}
hard negatives	{no, yes}
learning rate	$[10^{-4}, 10^{-1}]$
DGMC	
parameter	choices
ψ_1 / ψ_2 dimension*	$[32, 64, \ldots, 1024]$
ψ_1 / ψ_2 (number of) GCN layers*	{1, 2, 3, 4, 5}
ψ_1 / ψ_2 batch normalization*	{False, True}
ψ_1 / ψ_2 layer concatenation*	{False, True}
ψ_1 dropout	$[0.00, 0.05, \ldots, 1.0]$
ψ_2 dropout	0.0
trainable embeddings	False
(entity embedding) normalization	{never, always-l1, always-l2}
learning rate	$[10^{-3}, 10^{-1}]$
GCN-Align*	
parameter	choices
model output dimension	$[32, 64, \ldots, (embeddingdimension)]$
(number of) GCN layers	{1, 2, 3}
batch normalization	{False, True}
layer concatenation	{False, True}
final linear projection	{False, True}
dropout	$\{0.0, 0.1, \ldots, 0.5\}$
trainable embeddings	{False, True}
(entity embedding) normalization	{never, always-l1, always-l2}
(weight) sharing horizontal	{False, True}
learning rate	$[10^{-3}, 10^{-1}]$

4.1 Results

Table 4. Results in terms of H@1 for all investigated combinations of datasets, models and initializations. Each cell represents the *test* performance of the best configuration of hyperparameters chosen according to *validation* performance.

DBP15k (JAPE)

init	Wu			Xu		
subset	fr-en	ja-en	zh-en	fr-en	ja-en	zh-en
Zero Shot	79.47	63.48	56.07	83.70	65.64	59.40
GCN-Align*	81.81	67.45	57.94	86.74	67.65	60.32
RDGCN	86.91	**72.90**	66.44	86.82	74.35	**69.54**
DGMC	**89.35**	72.17	**69.98**	**90.12**	**76.60**	68.76

OpenEA

init	Sun			
subset	d-w	d-y	en-de	en-fr
Zero Shot	46.53	81.90	75.99	79.90
GCN-Align*	45.76	84.65	85.34	89.41
RDGCN	**64.28**	**98.41**	80.03	**91.52**
DGMC	51.29	88.60	**88.10**	89.40

WK3l15k

init	BERT	
subset	en-de	en-fr
Zero Shot	85.55	77.27
GCN-Align*	85.92	**78.22**
RDGCN	**86.76**	78.05
DGMC	84.08	73.92

Table 4 presents the overall results. We can observe several points.

Zero-Shot Performance Generally, there is an impressive Zero-Shot performance, ranging from 39.15% for `OpenEA d-w` to 83.85% `WK3l15k en-de`. Thus, even in the weakest setting, approximately 40% of the entities can be aligned solely from their label, without any sophisticated method. Consequently, this highlights that comparison against methods not using this information is unfair. For `DBP15k`, we can compare the initialization from Wu et al. [17], used, e.g., by RDGCN to the performance of the initialization by Xu et al. [7], used, e.g., by DGMC. We observe that Wu's initialization is 7-9% points stronger than Xu's initialization. For `OpenEA d-w` we obtain 39.15% zero-shot performance, despite the original labels of the `w` side being meaningless identifiers. This is only due to using attribute triples with a pre-defined set of "name" attributes, cf. Table 2.

Model Performance When comparing the performance of both analyzed models, we can observe that they have a clear advantage over both baselines in two of three datasets, but we cannot identify a single winner among them. Although the

performance of DGMC dropped compared to the results reported originally[10], it still leads by about 3-4 points on almost all DBP15k subsets. Therefore, it confirms our observation that a smaller test set automatically leads to better results. Furthermore, we can see that different initialization with entity name has also an effect on model performance, which especially applies to the ja-en subset for DGMC or fr-en for GCN-Align*. RDGCN has a clear advantage on the OpenEA subsets extracted from DBPedia with a margin of between 10 and 13 points on both subsets. Note that we significantly improved results of RDGCN on the OpenEA dataset through our extensive hyperparameter search in comparison with the original evaluation [15]. Interestingly, as can be seen in the next section, the main reason is *not* the exploiting of information about different relations. The WK3L15k dataset constitutes an interesting exception. The performance of the DGMC method, which is supposed to be robust against noise due to its correspondence refinement, is not better than the zero-shot results. While RDGCN and GCN-Align* can improve the results, the improvement by 1-2 points does not look very convincing. From these results, we conclude that there exists no silver bullet for the task of EA, and the method itself is still a hyperparameter. At the same time, we see that the most realistic dataset poses a real challenge for SotA methods.

4.2 Ablation: RDGCN

We additionally present the results of an ablation study for some model parameters of RDGCN on the OpenEA datasets in Table 5. For each presented parameter and each possible value, we fix this one parameter and select the best configuration among all configurations with the chosen parameter setting according to validation H@1. The cell then shows the *test* performance of this configuration. We highlight the best setting on the respective graph pair in bold font. Thus, all bold entries in one column are equal since they refer to the one best configuration according to validation performance. Note that these numbers also coincide with the performance reported in Table 4 for OpenEA. We make the following interesting observations: For all but one graph pair, *always normalizing* the entity representations before passing them into the layers is beneficial. For `d-y`, where this is not the case, the difference in performance is small. For the *number of GCN layers*, we observe an increase in performance from 0 to 2 layers, and on some datasets (`d-w`, `d-y`) even beyond. Thus, aggregating the entities' neighborhood seems beneficial, highlighting the importance of the graph structure. For the *number of interaction layers*, which perform *relation-aware* message passing, we observe that for two of the four subsets (`d-y`, `en-de`) the best configuration does not use any interaction layer. However, the difference is small. None of the best configurations uses *trainable node embeddings*. The *negative* l_1 *similarity* is superior on all datasets, with most of the others being close to it. Using the dot product seems to be sub-optimal, maybe due to its unbound value range. Regarding *hard negative mining*, there is no clear tendency,

[10] As a general rule, the results improve by 1-2 points when trained on train+validation and it is not going to change the picture.

Table 5. Ablation results for RDGCN on OpenEA datasets. The setting used by [17] is underlined. The first number is validation H@1, second number test H@1. Bold highlights the best configuration. Please notice that due to the specialties of EA evaluation, the test and validation performance are *not* directly comparable [3].

		subset			
parameter	value	d-w	d-y	en-de	en-fr
normalization	always	**84.06** / **64.28**	99.44 / 97.48	97.72 / **93.56**	**96.89** / **91.52**
	initial	82.67 / 62.58	**99.78** / 98.41	97.67 / 93.02	95.56 / 89.50
	never	78.39 / 61.77	99.72 / **98.53**	**98.11** / 80.03	95.44 / 90.14
GCN	0	57.33 / 50.79	92.33 / 83.83	**98.11** / 80.03	92.22 / 86.94
layers	1	73.33 / 56.66	99.33 / 98.15	96.00 / 91.63	94.50 / 90.49
	2	78.39 / 61.77	99.56 / 98.16	97.72 / **93.56**	**96.89** / **91.52**
	3	**84.06** / **64.28**	**99.78** / **98.41**	97.00 / 92.18	95.44 / 90.14
interaction	0	78.11 / 60.53	99.72 / **98.53**	97.72 / **93.56**	95.33 / 89.08
layers	1	78.39 / 61.77	**99.78** / 98.41	97.67 / 92.59	95.44 / 90.14
	2	82.67 / 62.58	99.56 / 98.16	**98.11** / 80.03	**96.89** / **91.52**
	3	**84.06** / **64.28**	99.50 / 97.85	97.67 / 93.02	95.56 / 89.50
trainable	no	**84.06** / **64.28**	99.72 / **98.53**	97.72 / **93.56**	**96.89** / **91.52**
embeddings	yes	82.67 / 62.58	**99.78** / 98.41	**98.11** / 80.03	95.56 / 89.50
similarity	cos	82.67 / 62.58	99.56 / 98.16	**98.11** / 80.03	95.56 / 89.50
	dot	63.28 / 40.80	91.50 / 79.81	85.17 / 78.54	89.94 / 78.17
	l1 (inv.)	77.89 / 60.78	99.50 / 97.85	93.78 / 88.96	94.06 / 88.69
	l1 (neg.)	**84.06** / **64.28**	99.72 / **98.53**	97.72 / **93.56**	**96.89** / **91.52**
	l2 (inv.)	75.28 / 60.20	96.72 / 92.06	95.06 / 90.13	94.44 / 89.60
	l2 (neg.)	72.50 / 51.04	**99.78** / 98.41	94.61 / 89.40	94.28 / 87.79
hard	no	82.67 / 62.58	**99.78** / **98.41**	**98.11** / 80.03	**96.89** / **91.52**
negatives	yes	**84.06** / **64.28**	99.67 / 98.30	97.72 / **93.56**	95.33 / 90.62

but considering the expensive calculation of the hard negatives (all-to-all kNN), its use might not be worthwhile. Another observation is that there sometimes is a huge gap between the test performance for the best configuration according to validation performance, and the best configuration according to test performance. For instance, if we had selected the hyperparameters according to test performance for `en-de`, we had obtained 93.53 H@1, while choosing them according to validation performance results in only 80.03 H@1 – a difference of 13.5% points. This emphasizes the need for a fair hyperparameter selection.

5 Conclusion

In this paper, we investigated state-of-the-art in Entity Alignment. Since we identified shortcomings in the commonly employed evaluation procedure including the lack of validation sets for hyperparameter tuning and different initializations, we provided a fair and sound evaluation for a wide range of datasets and initializations for two recent approaches and additional strong baselines. We also gave insights into the importance of individual components by an ablation study.

References

1. James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, 13:281–305, 2012.
2. Max Berrendorf, Evgeniy Faerman, Valentyn Melnychuk, Volker Tresp, and Thomas Seidl. Knowledge graph entity alignment with graph convolutional networks: Lessons learned. In Joemon M. Jose, Emine Yilmaz, João Magalhães, Pablo Castells, Nicola Ferro, Mário J. Silva, and Flávio Martins, editors, *Advances in Information Retrieval - 42nd European Conference on IR Research, ECIR 2020, Lisbon, Portugal, April 14-17, 2020, Proceedings, Part II*, volume 12036 of *Lecture Notes in Computer Science*, pages 3–11. Springer, 2020.
3. Max Berrendorf, Evgeniy Faerman, Laurent Vermue, and Volker Tresp. Interpretable and fair comparison of link prediction or entity alignment methods with adjusted mean rank. *CoRR*, abs/2002.06914, 2020.
4. Yixin Cao, Zhiyuan Liu, Chengjiang Li, Zhiyuan Liu, Juanzi Li, and Tat-Seng Chua. Multi-channel graph neural network for entity alignment. In *ACL (1)*, pages 1452–1461. Association for Computational Linguistics, 2019.
5. Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT (1)*, pages 4171–4186. Association for Computational Linguistics, 2019.
6. Laura Dietz, Chenyan Xiong, Jeff Dalton, and Edgar Meij. Special issue on knowledge graphs and semantics in text analysis and retrieval. *Inf. Retr. J.*, 22(3-4):229–231, 2019.
7. Matthias Fey, Jan Eric Lenssen, Christopher Morris, Jonathan Masci, and Nils M. Kriege. Deep graph matching consensus. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.
8. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society, 2016.
9. Chengjiang Li, Yixin Cao, Lei Hou, Jiaxin Shi, Juanzi Li, and Tat-Seng Chua. Semi-supervised entity alignment via joint knowledge embedding model and cross-graph model. In *EMNLP/IJCNLP (1)*, pages 2723–2732. Association for Computational Linguistics, 2019.
10. Xin Mao, Wenting Wang, Huimin Xu, Man Lan, and Yuanbin Wu. MRAEA: an efficient and robust entity alignment approach for cross-lingual knowledge graph. In *WSDM*, pages 420–428. ACM, 2020.
11. Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543. ACL, 2014.
12. Xiaofei Shi and Yanghua Xiao. Modeling multi-mapping relations for precise cross-lingual entity alignment. In *EMNLP/IJCNLP (1)*, pages 813–822. Association for Computational Linguistics, 2019.
13. Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *CoRR*, abs/1505.00387, 2015.

14. Zequn Sun, Chengming Wang, Wei Hu, Muhao Chen, Jian Dai, Wei Zhang, and Yuzhong Qu. Knowledge graph alignment network with gated multi-hop neighborhood aggregation. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 222–229. AAAI Press, 2020.
15. Zequn Sun, Qingheng Zhang, Wei Hu, Chengming Wang, Muhao Chen, Farahnaz Akrami, and Chengkai Li. A benchmarking study of embedding-based entity alignment for knowledge graphs. *Proc. VLDB Endow.*, 13(11):2326–2340, 2020.
16. Zhichun Wang, Qingsong Lv, Xiaohan Lan, and Yu Zhang. Cross-lingual knowledge graph alignment via graph convolutional networks. In *EMNLP*, pages 349–357. Association for Computational Linguistics, 2018.
17. Yuting Wu, Xiao Liu, Yansong Feng, Zheng Wang, Rui Yan, and Dongyan Zhao. Relation-aware entity alignment for heterogeneous knowledge graphs. In Sarit Kraus, editor, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 5278–5284. ijcai.org, 2019.
18. Yuting Wu, Xiao Liu, Yansong Feng, Zheng Wang, and Dongyan Zhao. Jointly learning entity and relation representations for entity alignment. In *EMNLP/IJCNLP (1)*, pages 240–249. Association for Computational Linguistics, 2019.
19. Hui Xu, Liyao Xiang, Youmin Le, Xiaoying Gan, Yuting Jia, Luoyi Fu, and Xinbing Wang. High-order relation construction and mining for graph matching. *CoRR*, abs/2010.04348, 2020.
20. Kun Xu, Liwei Wang, Mo Yu, Yansong Feng, Yan Song, Zhiguo Wang, and Dong Yu. Cross-lingual knowledge graph alignment via graph matching neural network. In Anna Korhonen, David R. Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 3156–3161. Association for Computational Linguistics, 2019.
21. Hsiu-Wei Yang, Yanyan Zou, Peng Shi, Wei Lu, Jimmy Lin, and Xu Sun. Aligning cross-lingual entities with multi-aspect information. In *EMNLP/IJCNLP (1)*, pages 4430–4440. Association for Computational Linguistics, 2019.
22. Rui Ye, Xin Li, Yujie Fang, Hongyu Zang, and Mingzhong Wang. A vectorized relational graph convolutional network for multi-relational network alignment. In *IJCAI*, pages 4135–4141. ijcai.org, 2019.
23. Qingheng Zhang, Zequn Sun, Wei Hu, Muhao Chen, Lingbing Guo, and Yuzhong Qu. Multi-view knowledge graph embedding for entity alignment. In *IJCAI*, pages 5429–5435. ijcai.org, 2019.
24. X. Zhao, W. Zeng, J. Tang, W. Wang, and F. Suchanek. An experimental study of state-of-the-art entity alignment approaches. *IEEE Transactions on Knowledge & Data Engineering*, (01):1–1, aug 5555.
25. Qiannan Zhu, Xiaofei Zhou, Jia Wu, Jianlong Tan, and Li Guo. Neighborhood-aware attentional representation for multilingual knowledge graphs. In *IJCAI*, pages 1943–1949. ijcai.org, 2019.