End-to-End Deep Entity Resolution without Labelled Instances

Franziska Neuhof[1], Marco Fisichella[1] and George Papadakis[2]

[1]L3S Research Center, Hannover, Germany
[2]National and Kapodistrian University of Athens, Athens, Greece

Abstract

Entity Resolution (ER) is a critical data integration task that is addressed through the Filtering-Verification framework. Filtering leverages unsupervised techniques for reducing the computational cost to the most similar candidate pairs, whereas Verification typically uses supervised learning techniques to distinguish between matching and non-matching pairs of entities. The state-of-the-art Verification methods leverage Deep Learning in combination with language models. Despite their high accuracy, their applicability is limited, due to their requirement for a large number of labelled instances from the dataset at hand. To address this issue, we propose a novel methodology for end-to-end Deep ER that waives the need for labelled instances. In essence, it associates the pair of data sources at hand $< \mathcal{D}_1, \mathcal{D}_2 >$ with the another pair of data sources of known groundtruth, $< d_1, d_2 >$, whose similarity distributions after Filtering are very close. The Verification method is then trained on the candidate pairs of $< d_1, d_2 >$ and applied to the candidate pairs of $< \mathcal{D}_1, \mathcal{D}_2 >$, linking their records without any labelled instances. Through a thorough experimental study that involves six datasets from two different domains, we demonstrate the high effectiveness of our approach, which outperforms established methods that require no labelled instances, while approximating the performance of the state-of-the-art Verification algorithms.

Keywords

Entity Resolution, Entity Matching, Deep Learning

1. Introduction

Entity resolution (**ER**) is a critical process in data management and analysis, aimed at identifying entities that correspond to the same real-world object across different data sources or within a single dataset. This process is essential for ensuring data quality, reducing redundancy, and enhancing the accuracy of analytics and decision-making. Traditionally, ER is implemented as a two-step process:

First, Filtering techniques identify entity pairs likely to match by using unsupervised methods. This reduces the search space complexity for the subsequent step to the most promising candidate pairs.

Second, Verification identifies the matching pairs out of the candidate pairs. Verification can also be split into multiple steps, Matching and Clustering. Traditional Matching techniques often rely on deterministic or probabilistic algorithms, which can be limited by their need for predefined rules or ground truth data. The recent advancements in machine learning (**ML**) and deep learning (**DL**) have opened new avenues for improving ER efficiency and accuracy, such as employing pre-trained language models. Clustering refines the matches generated by Matching by using again unsupervised methods.

However, most studies lack certain aspects that can hinder practical application: (i) Filtering and Verification are typically studied independently, even though Filtering determines the candidates available to Verification. The impact of the candidate set on training and testing performance is rarely analysed. (ii) ML and DL methods often require extensive labelled training data, which may not be available in many real-world scenarios. This limitation makes fine-tuning filtering techniques or training Deep Learning-based Matching algorithms challenging. (iii) Matching is often considered independently of Clustering, even though the latter is necessary for addressing any conflicts.

DOLAP 2026: 28th International Workshop on Design, Optimization, Languages and Analytical Processing of Big Data, co-located with EDBT/ICDT 2026, March 24, 2026, Tampere, Finland

franziska.neuhof@l3s.de (F. Neuhof); mfisichella@l3s.de (M. Fisichella); gpapadis@di.uoa.gr (G. Papadakis)
0009-0007-8294-7386 (F. Neuhof); 0000-0002-6894-1101 (M. Fisichella); 0000-0002-7298-9431 (G. Papadakis)

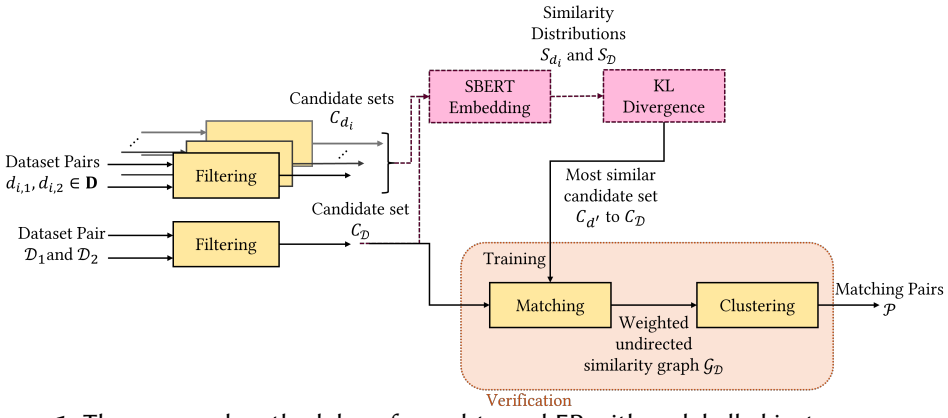

Figure 1: The proposed methodology for end-to-end ER with no labelled instances.

To address these challenges, we propose an end-to-end ER pipeline that integrates Filtering and Verification, which applies both Matching and Clustering. This pipeline is designed to be adaptable: The Matching algorithm can be trained on datasets with ground truth and then applied to different datasets, while Filtering and Clustering can be applied using default values, requiring no labelled instances. This allows the full pipeline to generalise effectively to datasets without labelled data, providing a robust foundation for entity resolution tasks across diverse datasets.

In this work, we focus on finding the overlap between two datasets, which are duplicate free. As shown in Figure 1, our approach identifies the pair of datasets from a set of dataset pairs $< d_{i,1}, d_{i,2} > \in \mathbf{D}$ with known ground truth which is most similar to the input pair of datasets without labelled instances, $< \mathcal{D}_1, \mathcal{D}_2 >$. We denote the dataset pair most similar to $< \mathcal{D}_1, \mathcal{D}_2 >$ by $< d'_1, d'_2 >$. This is achieved by comparing the similarity distributions after Filtering. The set of candidate pairs from $< d'_1, d'_2 >$ is then used to train the Matching algorithm, which is subsequently applied to the candidate pairs of $< \mathcal{D}_1, \mathcal{D}_2 >$. Finally, we employ clustering algorithms to refine the pairs generated by the Matching algorithm. Our approach aims to enhance the scalability and applicability of ER systems, making them more versatile and efficient in real-world applications. Through an extensive experimental study involving six datasets from two distinct domains, we demonstrate the high effectiveness of our approach. It approximates the performance of state-of-the-art Verification algorithms, which require labelled training instances, while outperforming established methods that require no labelled instances.

2. Problem Definition

We define as an *entity profile* with id i or just *entity* e_i as a uniquely identified set of name-value pairs: $e_i =< i, \{(\text{name}, \text{value})\} >$. Note that this plain data model accommodates data formats of any structuredness, from structured records (in relational databases or CSV files) to semi-structured entity descriptions (in RDF) and to free-text. A set of entity profiles of this form is called *data source*.

ER typically comes in one of two forms: (i) *Record Linkage*, also known as *Clean-Clean ER*, receives as input two individually duplicate-free data sources, $\mathcal{D}_1$ and $\mathcal{D}_2$, which are overlapping, i.e., sharing some of their entity profiles. (ii) *Deduplication*, also known as *Dirty ER*, receives as input a single dataset, $\mathcal{D}$, with duplicates in itself. In both cases, the output comprises the matching entity profiles.

In this context, we formally define the task we are tackling in this work as follows: *Given two data sources $< \mathcal{D}_1, \mathcal{D}_2 >$, which lack labelled entity pairs, along with a set of data source pairs $\mathbf{D}$, each with a known groundtruth, detect the matching entity profiles in $< \mathcal{D}_1, \mathcal{D}_2 >$ by leveraging the training instances provided by $\mathbf{D}$ so as to maximize the effectiveness in terms of recall, precision and F-Measure.*

This definition applies to Record Linkage, but can be easily extended to Deduplication. It focuses on end-to-end ER pipelines that perform both Filtering and Verification, as shown in Figure 1. Filtering receives two datasets, $\mathcal{D}_1$ and $\mathcal{D}_2$, and outputs the candidate pairs $\mathcal{C}$. Verification involves two steps: (i) Matching estimates the matching likelihood between the entities of each candidate pair. (ii) Clustering partitions the similarity graph into the set of matching entities, resolving the conflicts that may arise after Matching (e.g., an entity from $\mathcal{D}_1$ is matched with two or more entities from $\mathcal{D}_2$).

3. Approach

Our goal is to apply end-to-end ER pipelines that combine the state-of-the-art DL-based matching algorithms with an unsupervised state-of-the-art filtering method. We can waive the need to train the matchers on a large set of labelled instances by using other datasets with known-groundtruth as training sets. Based on Problem 1, which dictates that the input comprises a pair of data sources with no labelled instances, $< \mathcal{D}_1, \mathcal{D}_2 >$, along with a set of data source pairs $\mathbf{D}$, each with a known groundtruth, our solution consists of the following steps:

1) We apply a predetermined Filtering method with specific parameter configuration to each labelled data source pair $< d_{i,1}, d_{i,2} > \in \mathbf{D}$, yielding a set of candidate pairs C_{d_i}.

2) For each labelled data source pair $< d_{i,1}, d_{i,2} > \in \mathbf{D}$, we convert the entities e_i and e_j in each candidate pair $c_{i,j} \in C_{d_i}$ into the embedding vectors v_i and v_j using a pre-trained SBERT model. Then, we compute the cosine similarity of the resulting vectors, i.e., $cosine(v_i, v_j)$, yielding a similarity distribution S_{d_i} per data source pair $< d_{i,1}, d_{i,2} >$.

3) We apply the same filtering method with the same configuration to the unlabelled data source pair, $< \mathcal{D}_1, \mathcal{D}_2 >$, yielding the set of candidate pairs $C_{\mathcal{D}}$.

4) We estimate the similarity distribution $S_{\mathcal{D}}$, by converting all pairs of entities in $C_{\mathcal{D}}$ into pairs of SBERT embedding vectors and computing their cosine similarity.

5) Using KL-divergence, we estimate the similarity distribution $S_{d'}$ of the data source pairs in $\mathbf{D}$ that is closer to $S_{\mathcal{D}}$. The selected data source pair is denoted by d'.

6) We train the selected matching algorithm on all candidate pairs extracted from d', $C_{d'}$, using the respective groundtruth for assigning their labels.

7) We apply the trained matching algorithm on the candidate pairs of the given unlabelled data source pair, $C_{\mathcal{D}}$. As a result, each candidate pair in $C_{\mathcal{D}}$ is associated with a matching probability, yielding a weighted undirected similarity graph $\mathcal{G}_{\mathcal{D}}$.

8) We remove any conflicts from the matching outcomes, by applying a clustering algorithm to the similarity graph $\mathcal{G}_{\mathcal{D}}$ to produce the final output.

To put this methodology into practice, we need to specify the algorithms that are used for Filtering, Matching and Clustering.

Starting with the Filtering algorithm, we selected *kNN Join*, which converts each entity into a sparse q-gram vector and then identifies its k nearest neighbors using Cosine, Dice or Jaccard similarity. kNN Join consistently achieves very high effectiveness, while being easily configured [1]. Its default configuration actually exhibits the highest performance across all methods considered in [1]. Another advantage is its schema-agnostic functionality, which considers all attribute values, regardless of the associated attribute names. This way, it minimizes the need for human intervention, while increasing the robustness to schema heterogeneity and noise in attribute names and values.

For Matching, there is no clear winner among DITTO [2, 3], EMTransformer [4] and GNEM [5] in [6]. We combine our methodology with all three of them in order to experimentally identify the best choice.

Finally, for Clustering, the best balance between effectiveness and time efficiency is achieved by UMC, which excels in balanced datasets, where there is a relatively equal number of duplicates in $\mathcal{D}_1$ and $\mathcal{D}_2$, and EC, which excels in all other types of datasets [7].

3.1. Fine-tuning Configuration Parameters

The performance of the methods comprising our approach is significantly affected by their configuration parameters [1, 6, 7]. Hence, we consider two types of parameter configurations: (i) The *fine-tuned* ones, which correspond to the ideal case, where every method in the end-to-end ER pipeline has the optimal hyperparameter values for the data at hand. These values are determined through grid search. (ii) The *default* ones, which correspond to the realistic case, where the same hyperparameter values are used in all datasets. These values have been determined through extensive experimental analysis.

In this context, it is worth delving into the exact configuration parameters of each considered algorithm. kNN-Join is configured by: (i) `Prep`, a binary parameter that specifies whether pre-processing

Table 1
Statistics of the datasets used in our experiments. E_x stands for data source x, $|E_x|$ for its size, i.e., number of entities, and $|D|$ for the number of matching entities across the two data sources.

Dataset	E_1	$\|E_1\|$	E_2	$\|E_2\|$	$\|D\|$
D_1	Abt	1,076	Buy	1,064	1,064
D_2	Amazon	1,354	Google Products	2,935	1,102
D_3	IMDB	5,117	TMDB	6,042	1,965
D_4	IMDB	5,117	TVDB	5,178	966
D_5	TMDB	6,042	TVDB	5,178	1,035
D_6	Walmart	2,554	Amazon	22,068	853

Table 2
Configuration and performance of the selected filtering approach, kNN-Join, based on [1]. Note that $|C|$ stands for the number of candidate pairs generated by Filtering, Re for their recall, Pr for their precision and RT for the required run-time in seconds.

Dataset	Fine-tuned configuration						Fine-tuned performance				Default performance			
	Prep	Indexing	Q	Multiset	Similarity	K	$\|C\|$	Re	Pr	RT	$\|C\|$	Re	Pr	RT
D_1	true	largest	3	true	cosine	4	4,301	0.927	0.229	9.0	5,408	0.918	0.181	3.6
D_2	true	largest	5	true	cosine	26	35,339	0.889	0.028	29.4	6,856	0.781	0.126	26.1
D_3	false	smallest	5	false	cosine	1	6,352	0.969	0.300	149.8	31,771	0.828	0.051	62.5
D_4	false	smallest	5	false	cosine	1	5,389	0.957	0.171	97.1	31,295	0.898	0.028	28.5
D_5	false	smallest	5	false	cosine	1	5,629	0.970	0.178	113.2	28,644	0.992	0.036	44.2
D_6	true	largest	4	true	cosine	2	5,182	0.899	0.147	152.7	12,965	0.941	0.062	93.8

(i.e., stemming and stop-word removal) is applied or not, (ii) `Indexing`, a binary parameter that specifies whether the smallest data source is used for indexing and the largest one for querying or vice versa, (iii) Q, a numeric parameter q that determines the size of the q-gram, with $q \in [2, 5]$, (iv) `Multiset`, a binary parameter that specifies whether we consider a set of q-grams (ignoring duplicate ones) or a multi-set (treating each q-gram as a different signature), (v) `Similarity`, a categorical parameter that specifies the similarity function used to estimate how similar two sparse entity vectors are (i.e., Dice, Jaccard or cosine), and (vi) K, numeric parameter k that specifies the number of candidates per query entity, with $k \in [1, 10]$. The following default configuration achieves consistently high performance [1]: `Prep=true`, `Indexing= largest`, `k=5`, `Q=5`, `Similarity=cosine`, `Multiset=true`.

For the Matching algorithms, the number of epochs is the only hyperparameter that we fine-tune, with a maximum of 40 epochs. Preliminary experiments indicated that a suitable default value is 25 epochs – in many cases, the considered matching algorithms achieved their best performance with lower epochs, while a few reached their best performance after more than 30 epochs. For all other hyperparameters, we use default values determined by their open-source implementation.

For the Clustering algorithms, the only configuration parameter is the similarity threshold, $t \in [0, 1)$, which prunes the edges with a similarity score lower than t. The default value of t has been set to 0.5, as a matching probability lower than 0.5 indicates pairs of entities that most likely are not matching.

4. Experimental Analysis

Experimental Setup. The complete code (in Python) and data are available on GitHub[1]. We selected six commonly used datasets [6, 3, 4, 8, 9], from two distinct domains: D_1, D_2, and D_6 originate from the product domain, while D_3, D_4 and D_5 pertain to TV-Shows. The statistics of these datasets are reported in Table 1 in increasing computational cost for the brute-force approach ($|E_1 \times E_2|$). The experiments were conducted on a GPU cluster equipped with NVIDIA A100 GPUs with a memory of 40GB along with AMD EPYC CPUs. To ensure comparability, the experiments were restricted to using 1 GPU (if necessary) and 8 CPUs. Regarding the baseline methods, we use two kinds: (i) Approches

[1]https://github.com/still273/ertransfer

Table 3
The F-Measure of all considered methods across all datasets in Table 1. The labelled dataset for D_1 is D_2, for D_2 and D_6 it is D_1, for D_3 and D_5 it is D_4 and for D_4 it is D_3. In every dataset, the best baseline approach is highlighted in **red**, while the best of our approaches is highlighted in **blue**.

	Unique Mapping Clustering				Exact Clustering			
	Fine-tuned Configuration		Default Configuration		Fine-tuned Configuration		Default Configuration	
	Labelled Dataset	Same Dataset	Labelled Dataset	Same Dataset	Labelled Dataset	Same Dataset	Labelled Dataset	Same Dataset
DITTO	70.07%	88.20%	52.38%	85.71%	64.17%	85.35%	45.02%	85.71%
EMTransformer	67.92%	89.74%	**72.16%**	**91.60%**	65.68%	89.74%	69.87%	91.60%
GNEM	–	90.79%	70.92%	87.24%	–	87.38%	67.69%	87.24%
ZeroER	–	70.93%	–	0.00%	–	63.72%	–	0.00%
DADER	59.06%	–	59.69%	–	57.39%	–	58.82%	–
Unicorn	87.03%	–	82.65%	–	86.26%	–	81.87%	–

(a) Performance of the end-to-end pipelines on D_1.

DITTO	49.32%	**67.25%**	60.15%	66.28%	52.92%	66.27%	58.54%	66.28%
EMTransformer	41.88%	64.21%	51.65%	64.24%	42.78%	66.45%	50.87%	64.44%
GNEM	59.72%	–	**64.52%**	62.70%	55.60%	–	64.10%	62.67%
ZeroER	–	35.41%	–	5.23%	–	36.20%	–	5.23%
DADER	50.58%	–	49.75%	–	56.56%	–	50.00%	–
Unicorn	51.38%	–	57.47%	–	48.78%	–	56.87%	–

(b) Performance of the end-to-end pipelines on D_2.

DITTO	75.03%	88.50%	67.29%	91.69%	75.93%	89.67%	80.47%	91.69%
EMTransformer	**89.62%**	93.27%	82.87%	**93.29%**	89.52%	93.08%	82.87%	93.29%
GNEM	88.50%	92.50%	–	92.71%	88.65%	91.96%	–	92.71%
ZeroER	–	68.50%	–	0.15%	–	68.62%	–	0.15%
DADER	89.38%	–	72.59%	–	89.25%	–	72.59%	–
Unicorn	91.22%	–	89.67%	–	91.38%	–	90.17%	–

(c) Performance of the end-to-end pipelines on D_3.

DITTO	55.30%	80.50%	22.74%	75.00%	53.06%	80.62%	22.74%	75.00%
EMTransformer	**82.76%**	83.54%	77.53%	83.15%	81.72%	**84.15%**	77.59%	83.15%
GNEM	81.40%	82.13%	75.08%	81.54%	80.25%	80.73%	75.11%	81.54%
ZeroER	–	61.70%	–	55.90%	–	62.45%	–	55.90%
DADER	71.73%	–	74.60%	–	71.52%	–	74.60%	–
Unicorn	81.24%	–	80.80%	–	80.30%	–	81.15%	–

(d) Performance of the end-to-end pipelines on D_4.

DITTO	71.24%	81.13%	70.91%	81.60%	70.13%	81.19%	71.33%	81.60%
EMTransformer	72.03%	**86.83%**	67.99%	84.98%	**72.16%**	85.55%	68.64%	84.98%
GNEM	71.85%	81.25%	–	83.21%	71.53%	81.57%	–	83.21%
ZeroER	–	60.58%	–	60.39%	–	60.89%	–	60.39%
DADER	76.76%	–	63.16%	–	76.05%	–	63.16%	–
Unicorn	65.29%	–	65.45%	–	66.91%	–	66.08%	–

(e) Performance of the end-to-end pipelines on D_5.

DITTO	71.36%	**78.26%**	72.07%	70.25%	71.54%	74.40%	**73.00%**	70.25%
EMTransformer	69.76%	74.51%	63.63%	74.53%	71.02%	72.13%	64.82%	74.53%
GNEM	69.21%	67.63%	52.06%	68.66%	70.51%	66.95%	53.80%	68.66%
ZeroER	–	62.01%	–	66.27%	–	61.67%	–	66.27%
DADER	71.10%	–	56.19%	–	65.57%	–	56.75%	–
Unicorn	69.06%	–	67.57%	–	70.78%	–	67.74%	–

(f) Performance of the end-to-end pipelines on D_6.

specifically designed to be applied to datasets without labelled instances, namely ZeroER [10], Unicorn [11] and DADER [12, 13], and (ii) Traditional supervised learning matching algorithms, namely DITTO, EMTransformer and GNEM, in combination with our Filtering approach. These settings correspond to the ideal case, where the dataset at hand had labelled all candidate pairs generated by Filtering. In this case, we use 60% of the candidate pairs for training, 20% for validation and the remaining 20% for testing. Our methodology trains DITTO, EMTransformer and GNEM on the selected labelled dataset D', using 80% of the labelled candidate pairs as training data and the remaining 20% as validation data. Training is run for 40 epochs. Under the fine-tuned configuration settings, the learned model at the epoch with the highest validation F1 is applied to all candidate pairs of the unlabelled target dataset D.

Filtering performance. The performance of kNN-Join, is reported in Table 2 along with its fine-tuned and default parameter configurations. For the fine-tuned settings, we consider the configuration that

was determined in [1] through grid search with the following optimization goal: *maximize precision for recall at least 0.90; otherwise maximize recall.* In this way, Filtering ensures sufficiently high levels of recall for Verification, while minimizing the false positives, i.e., the non-matching candidate pairs. We observe that the overall Filtering time is quite low in all cases, as all datasets are processed in at most 2.5 minutes. Most importantly, the selected Filtering approach provides sufficiently high recall in almost all cases, at the cost of low precision. These settings call for a Verification method, which should maintain recall to very high levels, while raising precision by several times.

Research Question 1: *How does our methodology compare to ZeroER, DADER and Unicorn, the main end-to-end ER pipelines that also require no labelled instances?* Looking into Table 3, we observe that our methodology offers significant advantages over the other state-of-the-art approaches that require no labelled datasets. Combined with GNEM, it outperforms ZeroER, DADER and Unicorn, on average, provided that there is sufficient GPU RAM to support labelled datasets with large sets of candidate pairs. As an alternative, EMTransformer trades higher scalability and robustness for slightly lower effectiveness – only Unicorn achieves higher F-Measure, on average, because it requires a large amount of training data to achieve this performance on unlabelled datasets.

Research Question 2: *How does our methodology compare to the traditional DL-based settings, which require a large dataset of labelled instances of the dataset at hand?* We compare the performance on the "Labelled Dataset" columns of Table 3 with the adjacent ones of the "Same Dataset". DITTO's effectiveness raises significantly when trained on instances from the same dataset: it underperforms by 19% and 31% under the fine-tuned and the default configurations, respectively, regardless of the clustering algorithm. This applies to all datasets except D_6.

EMTransformer also underperforms the traditional settings in all cases, without any single exception (unlike DITTO). In half the datasets, though, its performance is very close to the traditional settings under the fine-tuned settings. The difference between the two approaches is $<$4%, $<$3% and $<$6% in D_3, D_4 and D_6, respectively, for both clustering algorithms. As a result, EMTransformer lies much closer to the traditional approach than DITTO.

The best performance, by far, is exhibited by GNEM. Under the fine-tuned configurations, it is very close to the traditional settings in D_3 and D_4, underperforming by just $<$4% and $<$1%, respectively, regardless of the clustering algorithm. In D_6, its performance even outperforms the traditional settings by 2% and 5% for UMC and EC, respectively. Only in D_5, it underperforms the traditional settings by $>$15%. However, it suffers from instability, given that it does not scale to D_1 and D_2. This applies to the default configurations, too, where it does not scale in D_3 and D_4.

Overall, GNEM constitutes the most effective choice for our approach in case there are no hardware limitations, exhibiting a performance very close or even better than the traditional settings. Te most robust solution for our approach is EMTransformer, at the cost of a slightly lower effectiveness.

5. Conclusions

We presented a novel methodology for applying deep ER end-to-end pipelines without requiring any labelled instances from the input data sources, $\mathcal{D}_1$ and $\mathcal{D}_2$. It leverages the Filtering-Verification framework, using a high-end unsupervised technique for the former step. The same Filtering technique is applied to a set of labelled dataset pairs $\mathbf{D}$, generating candidate pairs with high recall and low precision. The candidate pairs of $< \mathcal{D}_1, \mathcal{D}_2 >$ are compared with those of all dataset pairs in $\mathbf{D}$ and the one with the closest similarity distribution is used as a training set for the DL-based matching algorithm. The resulting trained matcher together with a clustering algorithm form the Verification step. Our experimental analysis demonstrates the high effectiveness of our methodology, which outperforms most relevant techniques from the literature. The best results are achieved when using GNEM as the matching algorithm, with EMTransformer following in close distance. In the future, we plan to investigate ways of improving the selection of the labelled dataset and perhaps of the instances extracted from it.

Acknowledgements. This work was partially funded by the EU Horizon project RECITALS (Grant Agreement 101168490).

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] F. Neuhof, M. Fisichella, G. Papadakis, K. Nikoletos, N. Augsten, W. Nejdl, M. Koubarakis, Open benchmark for filtering techniques in entity resolution, VLDB J. 33 (2024) 1671–1696.

[2] Y. Li, J. Li, Y. Suhara, J. Wang, W. Hirota, W. Tan, Deep entity matching: Challenges and opportunities, ACM J. Data Inf. Qual. 13 (2021) 1:1–1:17.

[3] Y. Li, J. Li, Y. Suhara, A. Doan, W. Tan, Deep entity matching with pre-trained language models, Proc. VLDB Endow. 14 (2020) 50–60.

[4] U. Brunner, K. Stockinger, Entity matching with transformer architectures - A step forward in data integration, in: A. Bonifati, Y. Zhou, M. A. V. Salles, A. Böhm, D. Olteanu, G. H. L. Fletcher, A. Khan, B. Yang (Eds.), Proceedings of the 23rd International Conference on Extending Database Technology, EDBT 2020, Copenhagen, Denmark, March 30 - April 02, 2020, OpenProceedings.org, 2020, pp. 463–473. URL: https://doi.org/10.5441/002/edbt.2020.58. doi:10.5441/002/EDBT.2020.58.

[5] R. Chen, Y. Shen, D. Zhang, GNEM: A generic one-to-set neural entity matching framework, in: WWW, 2020, pp. 1686–1694.

[6] G. Papadakis, N. Kirielle, P. Christen, T. Palpanas, A critical re-evaluation of record linkage benchmarks for learning-based matching algorithms, ICDE (2024) 3435–3448.

[7] G. Papadakis, V. Efthymiou, E. Thanos, O. Hassanzadeh, P. Christen, An analysis of one-to-one matching algorithms for entity resolution, VLDB J. 32 (2023) 1369–1400.

[8] H. Köpcke, A. Thor, E. Rahm, Evaluation of entity resolution approaches on real-world match problems, Proc. VLDB Endow. 3 (2010) 484–493.

[9] D. Obraczka, J. Schuchart, E. Rahm, Embedding-assisted entity resolution for knowledge graphs, in: (ESWC, volume 2873 of *CEUR Workshop Proceedings*, 2021.

[10] R. Wu, S. Chaba, S. Sawlani, X. Chu, S. Thirumuruganathan, Zeroer: Entity resolution using zero labeled examples, in: Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data, 2020, pp. 1149–1164.

[11] J. Tu, J. Fan, N. Tang, P. Wang, G. Li, X. Du, X. Jia, S. Gao, Unicorn: A unified multi-tasking model for supporting matching tasks in data integration 1 (2023).

[12] J. Tu, J. Fan, N. Tang, P. Wang, C. Chai, G. Li, R. Fan, X. Du, Domain adaptation for deep entity resolution, in: Proceedings of the 2022 International Conference on Management of Data, SIGMOD '22, Association for Computing Machinery, New York, NY, USA, 2022, p. 443–457. URL: https://doi.org/10.1145/3514221.3517870. doi:10.1145/3514221.3517870.

[13] J. Tu, X. Han, J. Fan, N. Tang, C. Chai, G. Li, X. Du, DADER: hands-off entity resolution with domain adaptation, Proc. VLDB Endow. 15 (2022) 3666–3669.